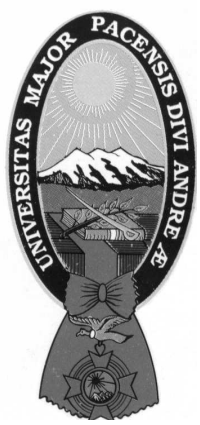
UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA ELECTRÓNICA



PROYECTO DE GRADO

DISEÑO DE UN PROTOTIPO DE SISTEMA NEURONAL DE DIAGNÓSTICO Y DETECCIÓN DE NEUMONÍA PULMONAR.

Proyecto de Grado presentado para optar al título de Licenciado en Ingeniería Electrónica.

POSTULANTE: **MARTIN MIGUEL JURADO CAMACHO**

TUTOR: **LUIS ALFONSO JURADO VISCARRA**

LA PAZ – BOLIVIA

2024

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE INGENIERIA

LA FACULTAD DE INGENIERIA DE LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) Visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) Copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) Copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la cita o referencia correspondiente en apego a las normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

AGRADECIMIENTOS.

A Dios.

A mis padres, Cristina y Rubén
por el apoyo que siempre me dieron y por ser
mis guías en todo el proceso de mi formación.

A mis docentes de la carrera de Ingeniería Electrónica,
especialmente a mi tutor de proyecto Alfonso Jurado
por las enseñanzas como profesional.

DEDICATORIAS.

Dedicado a mis hermanos Andrea, Matilde y Felipe
por ser mis más grandes ídolos.

Dedicado a toda mi familia que siempre
me ayudó incondicionalmente.

Y a mis amigos que me acogieron
como parte de su propia familia.

RESUMEN.

El proyecto presenta el diseño del prototipo de un sistema de clasificación de imágenes, a partir de una inteligencia artificial basada en redes neuronales y aprendizaje profundo, que permite realizar un diagnóstico de neumonía a partir de imágenes de radiografías de tórax de los pacientes.

El entrenamiento del sistema se realizó con la ayuda de un conjunto de datos de imágenes de rayos X de tórax, dicho set de datos fue obtenido del repositorio en línea llamado Kaggle, el cual ofrece distintos sets de datos para utilizar de forma libre en cualquier proyecto.

Las imágenes de rayos X de tórax fueron seleccionadas de pacientes pediátricos de uno a cinco años de edad del Centro Médico de Mujeres y Niños de Guangzhou, China. Las radiografías se examinaron inicialmente para el control de calidad eliminando todas las imágenes de baja calidad o ilegibles. Los diagnósticos de las imágenes fueron calificados por dos médicos expertos antes de ser aprobados para subir al repositorio de datos de Kaggle, las etiquetas de las imágenes pueden ser tres opciones: pacientes sanos, pacientes con neumonía bacteriana y pacientes con neumonía viral.

Todas las imágenes fueron pre procesadas para poder optimizar el entrenamiento de la red.

La red neuronal es del tipo Convolucional donde se utilizaron filtros de Convolución y filtros Max Pooling, además el diseño se basó en un modelo de Aprendizaje Supervisado, con funciones de activación Relu y Softmax, se contó con una función de coste basada en la Entropía Cruzada la cual fue optimizada mediante el algoritmo de Adam, y finalmente utilizando el algoritmo de Backpropagation se pudo implementar el Deep Learning sin problemas.

El prototipo neuronal fue entrenado, testeado y validado en un computador en la nube de Google, probando el diseño de diferentes arquitecturas y configuraciones de redes neuronales, hasta llegar a una red neuronal convolucional, que a partir de una imagen de radiografía de tórax permite detectar si un paciente presenta neumonía viral, neumonía bacteriana, o si no presenta ningún tipo de neumonía con una eficiencia del 87.55%.

INDICE

INDICE

INDICE DE FIGURAS.

INDICE TABLAS.

SECCION I. INTRODUCCION.

1 Capítulo 1 – Introducción.

1.1 Resumen.

En el presente proyecto se diseña el prototipo de un sistema de clasificación de imágenes realizado a partir de inteligencia artificial basada en redes neuronales y aprendizaje profundo.

El entrenamiento del sistema se realizó con la ayuda de un conjunto de datos de imágenes de rayos X de tórax de diferentes pacientes, dicho set de datos fue obtenido del repositorio en línea llamado Kaggle, el cual ofrece distintos sets de datos para utilizar de forma libre en cualquier proyecto.

Las imágenes de rayos X de tórax se seleccionaron de pacientes pediátricos de uno a cinco años de edad del Centro Médico de Mujeres y Niños de Guangzhou, China. Las radiografías se examinaron inicialmente para el control de calidad eliminando todas las imágenes de baja calidad o ilegibles. Los diagnósticos de las imágenes fueron calificados por dos médicos expertos antes de ser aprobados para subir al repositorio de datos de Kaggle, en el desarrollo del proyecto no se tomó en cuenta la presencia de un médico neumólogo que corrobore la clasificación de las imágenes, las etiquetas de las imágenes pueden ser tres opciones: pacientes sanos, pacientes con neumonía bacteriana y pacientes con neumonía viral.

El prototipo neuronal fue entrenado, testeado y validado con el objetivo de obtener una herramienta lógica, que pueda ser usada por un médico como apoyo en un diagnóstico eficaz de pacientes infectados por neumonía.

1.2 Antecedentes.

La Inteligencia Artificial ha sido desarrollada e implementada en gran medida en los últimos años, el campo de la medicina es uno de los más beneficiados por los desarrollos de esta disciplina, sus aplicaciones son numerosas y de muy alto beneficio, la inteligencia artificial se perfila como una herramienta capaz de analizar, procesar y aprender con rapidez enormes cantidades de información.

El avance a pasos agigantados de la tecnología en los últimos años ha generado cierto miedo por las personas a creer que las máquinas inteligentes lleguen a reemplazar el talento humano, lo cual es un pensamiento erróneo ya que todo el mundo debería aprender a utilizar la

inteligencia artificial en beneficio suyo. “La IA no te va a quitar tu trabajo, te lo quitará la persona que sepa usar la IA para desarrollar mejor tu trabajo.” (Jimenez, 2023)

1.3 Situación Actual.

Los sistemas inteligentes de visión artificial han evolucionado significativamente gracias a los avances en la tecnología, el desarrollo de los sistemas de computación y la creciente disponibilidad de datos. Hoy en día existe gran variedad de campos donde se aplican dichos sistemas, por ejemplo asistentes de chat inteligentes, reconocimiento de voz, tratamiento y generación de imágenes, etc.

Uno de los desarrollos más interesantes en los sistemas de clasificación inteligentes es la creciente importancia e implementación del "aprendizaje profundo" (deep learning). El aprendizaje profundo se basa en redes neuronales artificiales con múltiples capas, que son capaces de procesar grandes cantidades de datos y extraer patrones aún más complejos de ellos. Esto ha permitido mejoras significativas en áreas como el reconocimiento de imagen y de voz, el procesamiento del lenguaje natural y la toma de decisiones automatizada.

Se espera que estos sistemas inteligentes tengan un papel cada vez más importante en la automatización de tareas y la mejora de la eficiencia en todas las áreas posibles.

1.3.1 Sistemas Similares Desarrollados.

A lo largo del tiempo se realizaron proyectos de investigación y desarrollo de sistemas similares al propuesto, entre los cuales se pueden rescatar los más importantes:

CheXNet: desarrollado por investigadores de la Universidad de Stanford, Estados Unidos, en diciembre de 2017, el sistema consistía en una red neuronal convolucional de 121 capas que a partir de una imagen de rayos X del tórax, generaba la probabilidad de neumonía junto con un mapa de calor que localizaba las áreas de la imagen más indicativas de neumonía (Rajpurkar, y otros, 2017).

Kang Zhang, profesor de oftalmología en Shiley Eye Institute en San Diego (California, EEUU), en febrero de 2018 utilizó una red neuronal artificial para desarrollar un sistema que puede detectar rápidamente enfermedades oculares. Los científicos también probaron su herramienta para diagnosticar la neumonía infantil, descubrieron que el

programa podía diferenciar entre la neumonía viral y la bacteriana con una precisión superior al 90 por ciento (Zhang, 2021).

En Bolivia, desde junio de 2020, en el Hospital del Norte de la ciudad de El Alto, se aplica un sistema inteligente, desarrollado y donado por la empresa Huawei Bolivia, que permite ver en tiempo real los estudios tomográficos y radiográficos de lóbulos pulmonares de pacientes para determinar si presenta coronavirus o no (Viceministerio de Comunicación de Bolivia, 2020).

En el 2016, Linarez Ruiz Jimmy Héctor, desarrolló su proyecto de grado: Sistema de conteo de leucocitos en muestras de sangre humana por recuento diferencial automatizado - caso de estudio: aplicación de técnicas de procesamiento de imágenes capturadas por microscopios del laboratorio de análisis clínico del hospital de clínicas, La Paz Bolivia; en la Carrera de Ingeniería Electrónica de la Universidad Mayor de San Andrés.

En 2018, Layme Ordoñez, Roxana Paola, realizó su tesis: Clasificación de residuos plásticos mediante reconocimiento de imágenes, en la carrera de Informática de la Universidad Mayor de San Andrés.

Se puede destacar que la tecnología denominada visión por computador junto al aprendizaje automático de redes neuronales van de la mano en todas las aplicaciones mencionadas previamente, esto es posible gracias al gran desarrollo de la tecnología de los sistemas de computación actualmente implementado como una mención dentro de la carrera de Ingeniería Electrónica de la Universidad Mayor de San Andrés, dando lugar a diseñar e implementar diferentes sistemas que permitan realizar grandes procesos de manera mucho más rápida y eficiente.

1.3.2 Tendencias Tecnológicas.

Es importante recalcar dos modelos desarrollados por Open AI (organización de desarrollo e investigación de Inteligencia Artificial), los cuales son el presente y pueden ser el futuro de la inteligencia artificial.

El modelo de lenguaje de OpenAI conocido como GPT (Generative Pre-trained Transformer) fue lanzado en junio de 2020. Es el modelo de lenguaje más grande y avanzado que OpenAI ha creado hasta ahora, con 175 mil millones de parámetros. Desde

su lanzamiento, ha sido utilizado en una amplia variedad de aplicaciones, desde la generación de texto para chatbots hasta la creación de contenido web y la investigación en áreas como la robótica y la ciencia de datos (OpenAI, GPT-4 Technical Report, 2023).

El Procesamiento de Lenguaje Natural (NLP) es una rama de la inteligencia artificial que se centra en la interacción entre las computadoras y el lenguaje humano. Chat GPT está desarrollado en base a NLP el cual se enfoca en desarrollar algoritmos y técnicas que permitan a las computadoras entender, interpretar y generar lenguaje humano en forma natural. Es por eso que NLP aborda varios aspectos del lenguaje humano, incluyendo la comprensión del significado de las palabras, la sintaxis de las frases y la estructura de los párrafos y textos completos.

"DALL-E 2" ("Dali-Disguise Recognition") lanzado en 2021. Es un modelo de aprendizaje profundo que utiliza la tecnología de Redes Generativas Adversarias (GAN) para generar imágenes a partir de descripciones de texto, lo que lo hace capaz de crear imágenes complejas y detalladas que no existen en la vida real. Este modelo tiene la capacidad adicional de reconocer y diferenciar objetos en una imagen y generar imágenes a partir de descripciones más detalladas y complejas. También es capaz de realizar tareas más avanzadas, como el relleno de espacios en blanco en imágenes existentes y la creación de escenas completas a partir de descripciones de texto (OpenAI, DALL-E2, 2022).

El desarrollo de DALL-E 2 es un ejemplo del rápido avance de la tecnología de aprendizaje automático y la inteligencia artificial, y tiene grandes potenciales en una amplia variedad de aplicaciones, desde el diseño gráfico y la creación de contenido digital hasta la recreación de imágenes.

La unión de los dos sistemas inteligentes mencionados es una tendencia a futuro para lograr cada vez sistemas más completos que puedan realizar múltiples tareas sin dificultades.

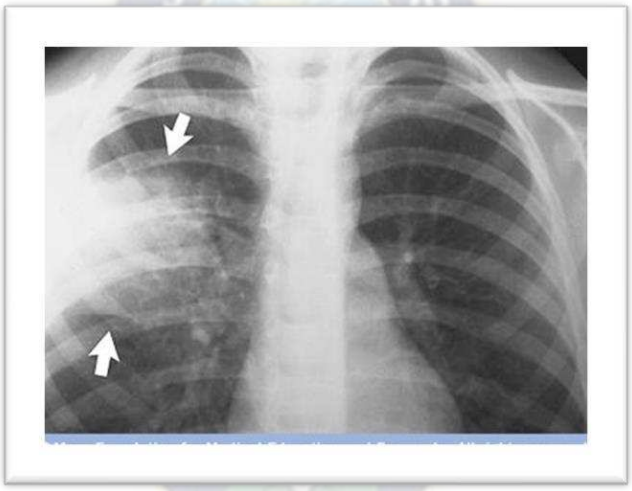
1.4 Planteamiento del Problema.

La Neumonía es una infección en los pulmones, causada por el contagio de un virus o una bacteria, dicha infección es una de las principales causas de muerte en todo el mundo, especialmente en los países en desarrollo.

El diagnóstico de neumonía debe ser realizado por un médico especialista tomando en cuenta tanto los síntomas cardinales como tos, fiebre y dolor pleurítico (dolor en el tórax), como también un análisis de sangre o una radiografía de tórax donde se aprecie la infección pulmonar.

Figura 1

Radiografía de tórax.



Nota. Muestra una zona de inflamación del pulmón. Adaptado de *Chest X-ray showing pneumonia*, de Mayo Foundation for Medical Education and Research (MFMER), Sitio web: https://www.mayoclinic.org/diseases-conditions/pneumonia/multimedia/chest-x-ray-showing-pneumonia/img-20005827.

En el mes de junio de 2023, el Servicio Departamental de Salud, SEDES La Paz, informó que se presentó un incremento de IRA´s (Infección respiratoria aguda), alcanzando a más de 19 mil casos, de los cuales 696 cuadros son de Neumonía.

En la nota desarrollada por SEDES, el Dr. Prisley Riveros afirma que los casos de fallecimiento se deben al descuido de los padres que no llevan a sus niños a los Centros de Salud ante la presencia de los primeros síntomas. SEDES informó que el incremento de los casos de neumonía en comparación del mes de mayo de 2023, alcanzó el 12%. (SEDES, 2023).

En el occidente de Bolivia, donde la temperatura es más baja, aumentan los casos de infecciones respiratorias agudas.

Según reportes epidemiológicos, la neumonía es una de las primeras causas de mortalidad en varios países latinoamericanos en niños menores de 5 años y su tendencia es ascendente en los últimos tres años. Situación en la cual los servicios de salud han puesto énfasis durante los últimos años, movilizando personal de salud hacia las zonas alejadas de las ciudades para realizar los controles (Organización Mundial de La Salud, 2022).

En la actualidad en Bolivia, no existe un sistema libre que pueda ser usado en cualquier computador que permita al médico realizar diagnósticos con eficiencia y eficacia para determinar esta infección mortal pulmonar.

En áreas rurales de los diferentes departamentos de Bolivia no existen los especialistas necesarios que puedan realizar un diagnóstico preciso y oportuno a pacientes con síntomas sospechosos.

1.5 Objetivos.

1.5.1 Objetivo General.

Diseñar el prototipo de un sistema de computación, basado en inteligencia artificial y redes neuronales, que permita realizar un diagnóstico de neumonía a partir de imágenes de radiografías de tórax de los pacientes.

1.5.2 Objetivos Específicos.

- Utilizar Deep Learning para obtener mejores resultados.
- Realizar el preprocesamiento de imágenes necesario para mejorar el entrenamiento del sistema.
- Optimizar al máximo el algoritmo de entrenamiento, en base a testeo y validación para mejorar el aprendizaje de la red neuronal.
- Determinar los niveles de confiabilidad, verificar el aprendizaje del sistema y no desarrollar un sistema experto.

1.6 Justificación.

1.6.1 Justificación Social.

El prototipo propuesto dará lugar a un sistema inteligente dedicado al diagnóstico de neumonía, lo cual significará una gran ayuda en la detección oportuna de dicha enfermedad, tanto en las ciudades como en las áreas rurales ya que se tratará de un sistema de libre acceso para ser utilizado en cualquier computador con cualquier sistema operativo, lo que no significa de ninguna manera reemplazar al conocimiento del médico especialista, más bien, se trata de un sistema que apoye en la toma de decisiones.

1.7 Alcances y Limitaciones.

El sistema será capaz de identificar si el paciente presenta neumonía viral, o neumonía bacteriana o si no presenta ningún tipo de neumonía.

El proyecto se realizará en lenguaje Python, tomando como herramientas diferentes librerías dedicadas al desarrollo de machine learning y aplicación de Deep learning.

Se desarrollará un prototipo el cual puede funcionar en un computador junto a una imagen de rayos X del paciente, el sistema no será capaz de tomar la radiografía al paciente.

El set de datos obtenido consta de imágenes de rayos X de pacientes pediátricos de uno a cinco años, por lo que el sistema solo será utilizado en pacientes en ese rango y edad.

En el desarrollo del proyecto no se realizará un estudio de factibilidad económica ya que no se requiere de recursos significativos para el diseño.

SECCION II. INVESTIGACIÓN.

SECCION II. INVESTIGACIÓN.

2 Capítulo 2 – Marco de Referencia.

2.1 Descripción del Escenario de Trabajo.

Clínicas o centros de salud de áreas rurales del país, son el principal objetivo en donde puede ser aplicado el sistema, ya que no será necesario de contar con conexión a internet ni computadores de alto rendimiento para el uso del mismo.

También en las ciudades, en cualquier hospital, el sistema inteligente de detección de neumonía, puede ser un gran apoyo a cualquier médico general en el diagnóstico de neumonía de forma temprana y oportuna.

3 Capítulo 3 – Marco Teórico.

3.1 Conceptos Esenciales de la Imagen Rayos X.

Los rayos X son una forma de radiación electromagnética que en medicina se utilizan para producir imágenes de estructuras internas del cuerpo humano, como huesos y órganos. La técnica de la radiografía convencional utiliza una fuente de rayos X que emite radiación hacia el cuerpo, y la radiación que atraviesa el cuerpo se detecta en una placa radiográfica o en un detector digital. Las estructuras densas, como los huesos, absorben más radiación que los tejidos blandos, lo que permite la creación de una imagen que muestra las diferentes densidades en el interior del cuerpo, dicha imagen puede ser almacenada en diferentes formatos para ser visualizada desde un computador.

3.2 Neumonía.

La neumonía es una inflamación aguda de los pulmones que puede ser causada por una infección viral, bacteriana u fúngica (hongos). Los síntomas comunes incluyen fiebre, tos con flema, dificultad para respirar y dolor en el pecho. Esta enfermedad puede afectar a personas de todas las edades, pero es más común en los niños y en los adultos mayores.

El diagnóstico de la neumonía se basa en los síntomas del paciente y en los resultados de las pruebas de diagnóstico, como una radiografía de tórax y un análisis de sangre.

Las pruebas más comunes que se utilizan para diagnosticar la neumonía incluyen:

Radiografía de tórax: es una prueba de imagen que utiliza rayos X para tomar imágenes de los pulmones y puede mostrar la presencia de líquido o inflamación en los pulmones.

Análisis de sangre: se pueden realizar pruebas de sangre para buscar signos de infección, como un aumento en el número de glóbulos blancos.

Cultivo de esputo: se recoge una muestra de flema de los pulmones para buscar la presencia de bacterias u hongos que puedan estar causando la neumonía.

Tomografía computarizada de tórax: se utiliza para obtener imágenes más detalladas de los pulmones y para detectar cualquier complicación de la neumonía.

Prueba de oxígeno en sangre: se mide la cantidad de oxígeno en la sangre para evaluar la capacidad del cuerpo para absorber y distribuir el oxígeno adecuadamente.

Examen físico: durante el examen físico, el médico escucha los sonidos respiratorios del paciente con un estetoscopio y busca signos de inflamación o infección en los pulmones.

El diagnóstico preciso de la neumonía es esencial para determinar el mejor tratamiento y prevenir complicaciones graves.

3.3 Inteligencia Artificial.

La inteligencia humana es una capacidad general de cada persona de poder razonar, pensar, aprender, sentir y muchas cosas más. Es lo que diferencia a las personas de los animales, si bien los animales presentan su propia forma de inteligencia, los seres humanos presentan un nivel de inteligencia mucho mayor a cualquier otra especie.

Por otra parte, la Inteligencia Artificial (IA) ha sido definida por muchas personas a su manera, debido a su complejidad, la IA no tiene una definición exacta establecida, pero se podría acercar como: “La capacidad de una máquina de imitar los comportamientos inteligentes de los seres humanos.” (Morandín-Ahuerma, 2023).

La IA puede ser débil o fuerte según la cantidad de tareas que puede realizar una máquina, por ejemplo, una IA es débil, cuando únicamente puede realizar una sola tarea, aunque lo haga a un nivel muy superior al de un ser humano, como por ejemplo jugar ajedrez.

Por lo contrario, una IA fuerte puede realizar diferentes tareas de forma super dotada, jugar ajedrez, conducir un automóvil, correr, etc. Al día de hoy no existen IA fuertes, pero si existen IA que se acercan mucho por ejemplo como el modelo GPT, una IA del área del Natural Lenguage Procesing (NLP), que únicamente ha sido entrenada para generar texto, pero al realizar pruebas en ella se puede verificar que la IA además puede traducir diferentes idiomas, resumir texto, hasta programar partes de código en diferentes lenguajes.

3.4 Aprendizaje Automático.

Existen diferentes áreas de la IA, como por ejemplo la robótica, el NLP, generación de imágenes, la conversión de voz a texto o de texto a voz, la visión por computador y muchas más, el desarrollo de este proyecto se basa en el área del Aprendizaje Automático o Machine Learning (ML).

El Aprendizaje Automático es una parte de la IA que busca la creación de sistemas con la capacidad de aprender bajo distintas circunstancias.

Aprender es generalizar el conocimiento a partir de ciertas experiencias, lo cual es exactamente lo que se quiere lograr en una máquina, que aprenda y obtenga conocimiento sobre la realización una tarea en base a su experiencia bajo diferentes circunstancias de la misma.

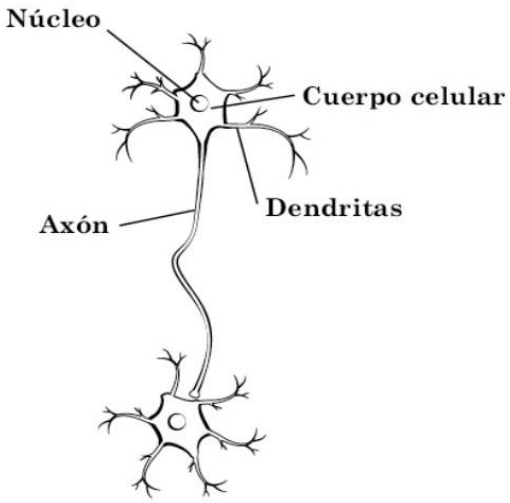
3.5 Redes Neuronales Artificiales.

Una red neuronal artificial es un modelo computacional que intenta imitar el comportamiento de las neuronas en el cerebro humano, la red está formada por neuronas y capas, existe diferentes tipos de estructuras de redes neuronales que permiten mejores o peores resultados en el momento del aprendizaje.

Las neuronas en el cerebro humano tienen las características mostradas en la figura 2, la información la reciben por las Dendritas y el Axón es la salida de la neurona, la conexión entre dos neuronas se denomina sinapsis.

Figura 2

La neurona.



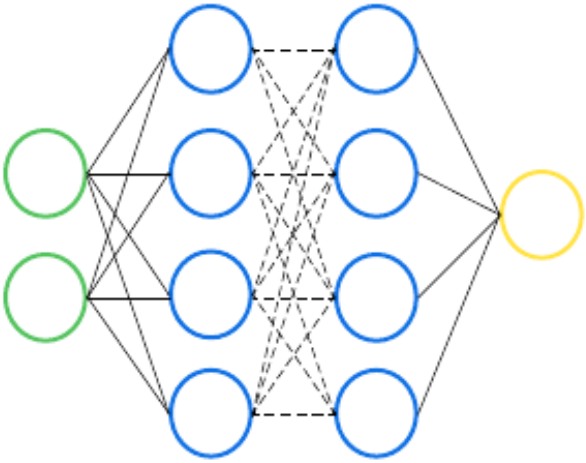
Nota. Adaptado de *La biología de la memoria, Eric R. Kandel Sitio Web: https://reader.digitalbooks.pro/content/preview/books/35552/book/OEBPS/1__El_contador_de_historias68989.html.*

Las neuronas reaccionan a pequeños impulsos eléctricos, también llamados impulsos nerviosos, recibidos por las Dendritas y generan una salida por el Axón, la conexión de millones de neuronas en el cerebro da lugar a los comportamientos inteligentes de las personas, como por ejemplo recuerdos, pensamientos, e incluso sueños.

De la misma manera, la arquitectura de las redes neuronales artificiales es conformada por muchas neuronas las cuales se organizan en capas, la conexión entre las neuronas artificiales se realiza mediante enlaces, como se muestra en la figura 3, donde las neuronas verdes son las entradas que reciben la información, las azules son las ocultas, que contienen cálculos intermedios de la red, y las amarillas son las salidas que contienen el resultado.

Figura 3

Esquema de una red neuronal Artificial.



Nota. Elaboración propia en lucid.app.

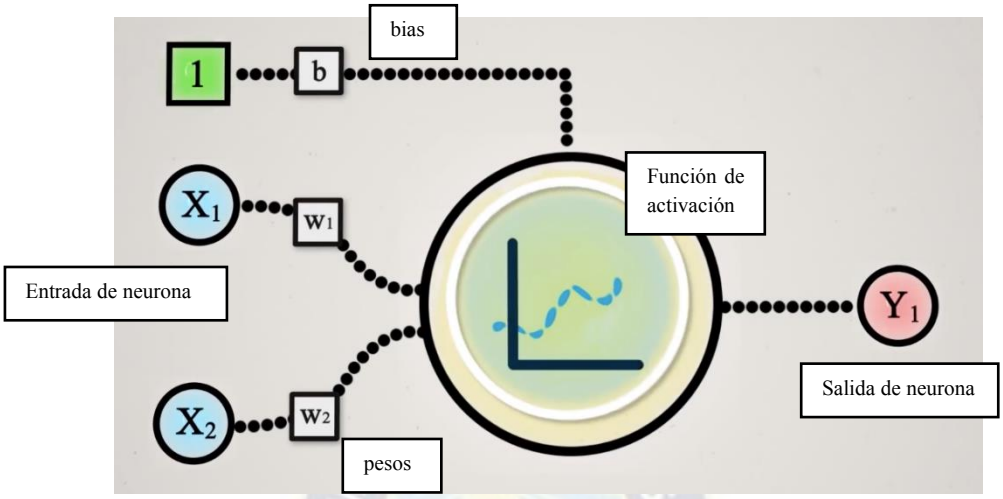
No existe un límite para las capas ocultas, depende mucho de la tarea que se requiera realizar, el número de neuronas de la capa de entradas así como las de la salida se tienen que calcular en base a la información que se va a procesar en la red.

3.5.1 La Neurona.

En cada neurona se realiza una operación matemática, observando la figura 4, se incluyen las entradas Xi, los pesos de los enlaces Wi, el termino de bias b (sesgo), y se realiza una sumatoria de todos los elementos, en donde si se varia cada parámetro se obtiene una ecuación de una recta que va cambiando en el espacio, dicha recta puede ser modificada gracias a una función de activación f(x) que introducirá un componente no lineal a la neurona, para dar paso a representar sistemas más complejos.

Figura 4

Componentes de una neurona.



Nota. Adaptado de *¿Qué es una Red Neuronal? Parte 1: La Neurona*, Dot CSV, 2019, YouTube:https://www.youtube.com/watch?v=uwbHOpp9xkc&list=PLOgd76BhmcC_E2RjgIIJZd1DQdYHcVf0&index=7.

Con toda la información recibida, la neurona genera la siguiente función de salida:

$$Z = W_1 * X_1 + W_2 * X_2 + b$$

La ecuación de la recta no es conveniente para representar funciones complejas, es por eso que se hace uso de la función de activación:

$$Y_1 = F(Z)$$

De esta manera la neurona obtiene su salida lista para ser conectada al peso de los enlaces de las siguientes neuronas.

Las Redes Neuronales son capaces de aprender de forma jerarquizada, es decir la información se aprende por niveles, donde las primeras capas aprenden conceptos muy concretos y en las capas posteriores se usa la información aprendida previamente para aprender conceptos más abstractos.

Mientras se agreguen más capas ocultas a la red neuronal, se convierte en un algoritmo más complejo, este incremento en las capas y en la complejidad es lo que da paso a convertirse en un algoritmo de Deep Learning (Aprendizaje Profundo).

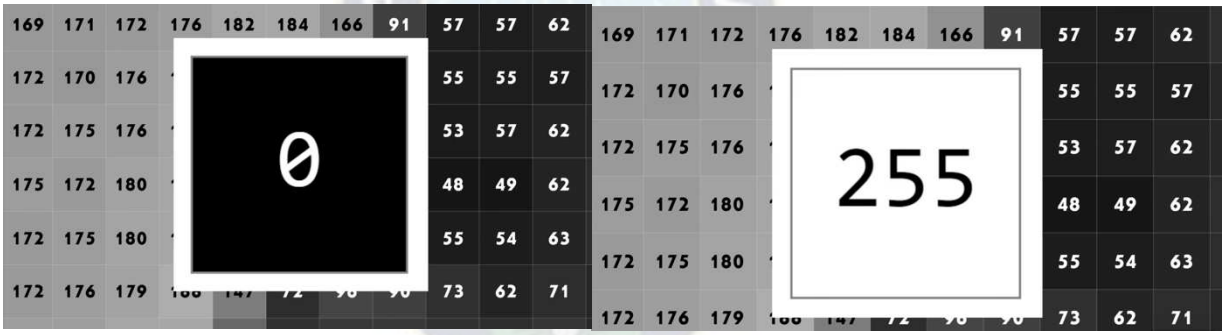
3.5.2 Red Neuronal Convolucional.

Una red neuronal convolucional (RNC) es un tipo de red neuronal especializada en procesar datos de imágenes. Está compuesta por capas de convolución y pooling, que son capaces de extraer características específicas de la imagen para realizar tareas como la clasificación o el reconocimiento de objetos.

Para introducir imágenes a una red neuronal, es conveniente transformarlas en escala de grises, así cada pixel puede ser representado como un único número de 0 a 255, y ya no en tres dimensiones RGB (red, green, blue) como es habitual, en la siguiente figura se muestra la representación del pixel en escala de grises.

Figura 5

Representación del pixel en Escala de Grises.



Nota. Adaptado de *Reconocimiento de imágenes con IA Convoluciones y filtros*, Ringa Tech, 2021, YouTube: https://www.youtube.com/watch?v=AwTH_0yW9_I.

Las redes neuronales convolucionales son muy útiles para tareas de reconocimiento de imágenes, clasificación de objetos y detección de objetos en tiempo real. Son utilizadas en una variedad de aplicaciones prácticas, como en sistemas de seguridad, vehículos autónomos, reconocimiento de caras, entre otros.

3.5.3 Filtros de Convolución.

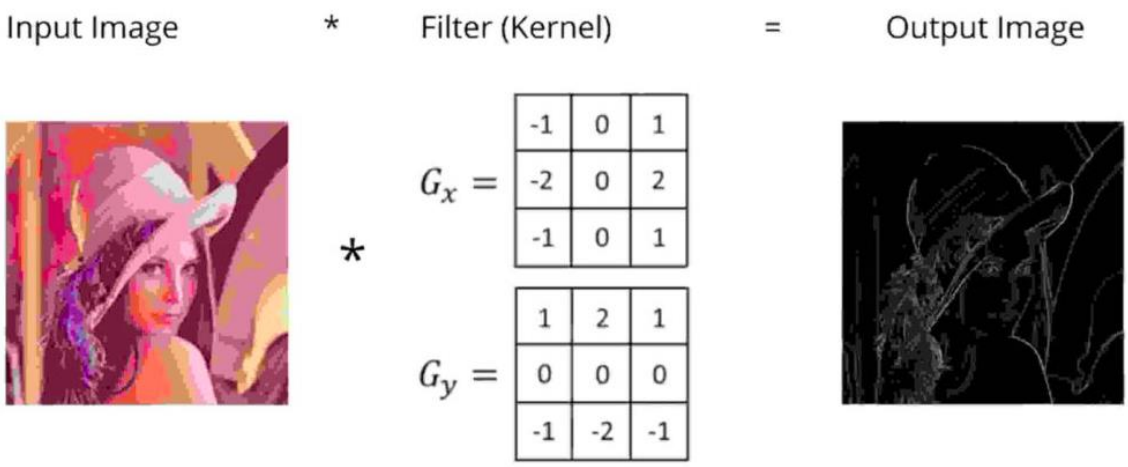
Una imagen ingresa a la red neuronal convolucional por la capa de entrada compuesta por filtros que permiten procesar la imagen y extraer características importantes, como bordes, esquinas y formas geométricas. Los filtros, también conocidos como "kernels"

en inglés, realizan una operación matemática conocida como convolución sobre la imagen de entrada, creando una "imagen convolucionada" que resalta las características de la imagen.

En la figura 6 se puede observar un ejemplo de convolución de una imagen, el filtro G_y permite resaltar los segmentos horizontales de la imagen mientras que el filtro G_x resalta los segmentos verticales, la combinación de los dos filtros da como resultado la imagen de la derecha en donde se observa una segmentación de todos los bordes de la imagen real mostrando las partes importantes de la misma.

Figura 6

Aplicación de filtros de convolución.



Nota. Adaptado de *Redes neuronales de convolución en pocas palabras Guía de la A a la Z*, Sandesh Kumar, 2020, https://www.herevego.com/redes-neuronales-de-convolucion/

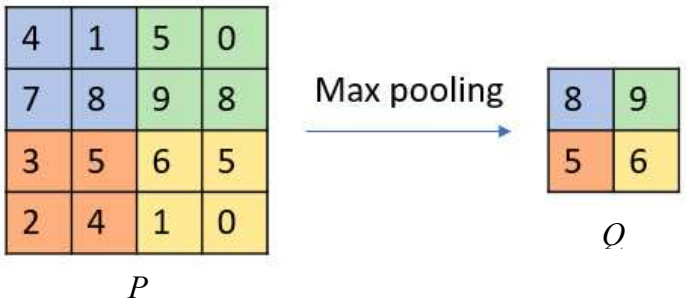
3.5.4 Pooling.

La capa de pooling reduce el tamaño de la imagen convolucionada, disminuyendo la cantidad de datos que se deben procesar en capas posteriores. El pooling se realiza utilizando diferentes técnicas, como el "max pooling" que selecciona el valor máximo en un área específica de la imagen, o el "average pooling" que calcula el promedio de los valores en un área.

A continuación, se muestra una matriz *P* en la cual si se aplica max pooling se obtiene la matriz más pequeña *Q*, se puede verificar que *Q* está compuesta por los valores máximos de los cuadrantes diferenciados por colores de *P*.

Figura 7

Representación Max Pooling.



P *Q*

Nota. Adaptado de *Visión artificial con redes convolucionales (CNN)*, Alberto García Serrano, 2019, https://www.ellaberintodefalken.com/2019/10/vision-artificial-redes-convolucionales-CNN.html

Después de múltiples capas de convolución y pooling, se utilizan capas totalmente conectadas para realizar la tarea específica de la red neuronal, como la clasificación de imágenes en diferentes categorías.

3.6 Modelos de Aprendizaje.

El Aprendizaje Automático tiene como objetivo crear sistemas con la capacidad de aprendizaje, es decir poder generalizar el conocimiento a partir de un conjunto de experiencias.

En esta área existen diferentes modelos de aprendizaje los cuales pueden ser:

3.6.1 Aprendizaje Supervisado.

Un modelo de aprendizaje supervisado cuenta con un data set donde cada dato se encuentre correctamente clasificado en una categoría, de esta forma el sistema realiza el entrenamiento observando y comparando la salida de la red, es decir la predicción, con la salida real esperada.

En este tipo de modelos se utilizan diferentes tipos de funciones de coste para comparar las salidas y minimizar el error generado.

El aprendizaje Supervisado se basa en descubrir la relación existente entre una variable de entrada y una variable de salida deseada, tras muchos ejemplos el algoritmo será capaz de dar un resultado correcto incluso cuando se le introduzca valores que no haya visto antes.

3.6.2 Aprendizaje No Supervisado.

La diferencia en este modelo se encuentra en el data set de entrenamiento, en donde los datos no están clasificados en categorías, entonces es tarea del sistema realizar una clasificación de los datos por sí solo agrupando clústeres, dicho proceso se lo logra gracias a diferentes algoritmos como k-means entre otros.

Después del entrenamiento, el sistema inteligente logrará realizar su predicción en base a su propia clasificación aprendida.

3.6.3 Aprendizaje por Refuerzo.

El entrenamiento del modelo se basa en recompensas o castigos que obtiene la red mientras va ajustando sus parámetros, este modelo es utilizado en sistemas inteligentes desarrollados para jugar juegos o sistemas inteligentes que den lugar a la conducción autónoma de un automóvil.

Generalmente estos sistemas tienen que ser entrenados en una alta cantidad de épocas, en algunas ocasiones se utiliza un agente con mucho menos entrenamiento para que compita contra sí mismo, es decir el mismo agente, pero con más épocas de entrenamiento.

3.7 Función de Activación.

Las funciones de activación son funciones que introducen no linealidad a las capas de una red neuronal, la no linealidad permite que la red neuronal pueda aprender funciones más complejas y representar patrones más sofisticados de los datos de entrada.

El trabajo de las neuronas consiste en formar una ecuación de una recta con los parámetros que obtiene a su entrada, de esta manera conectando muchas neuronas solo se obtendrá ecuaciones

de recta para representar funciones dando como resultado malas predicciones y malos resultados de la red.

Es por eso que se utiliza funciones de activación para que la red neuronal pueda representar funciones mucho más complejas en planos de varias dimensiones.

Algunas funciones de activación son las siguientes:

3.7.1 Función Sigmoide.

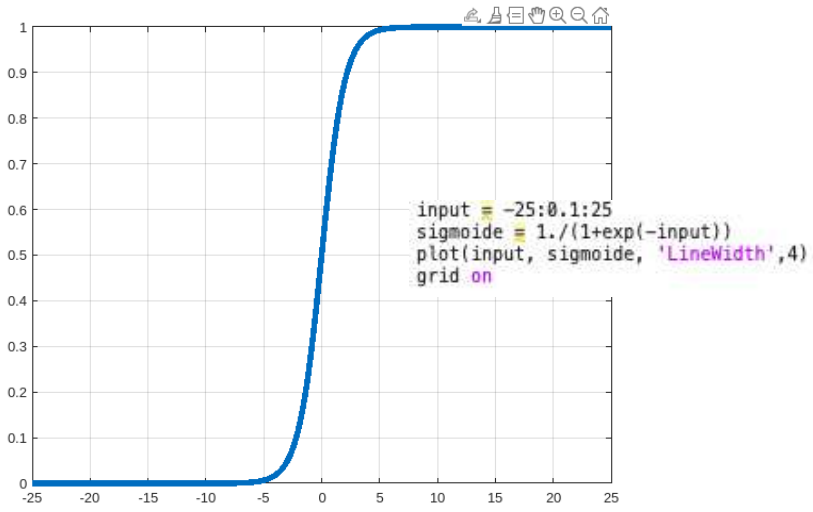
La función sigmoide se define como:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (1)$$

Graficada en la siguiente figura; devuelve valores entre 0 y 1, puede ser útil para problemas de clasificación binaria simples.

Figura 8

Función Sigmoide.



Nota. Elaboración propia en Matlab.

3.7.2 Función ReLU.

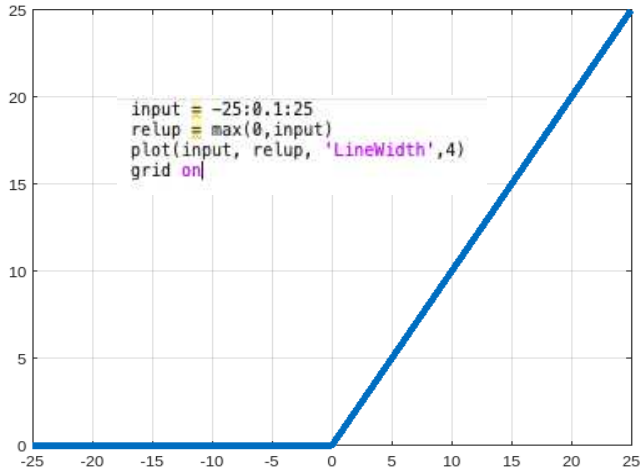
La función ReLU (unidad lineal rectificada) se define como:

$$f(x) = \max(0, x) \qquad (2)$$

Devuelve valores de 0 para cualquier entrada negativa y la misma entrada para cualquier valor positivo. Es una de las funciones de activación más populares en la actualidad, debido a su simplicidad y eficacia en la mayoría de los casos.

Figura 9

Función ReLu.



Nota. Elaboración propia en Matlab.

3.7.3 Función Tangente Hiperbólica.

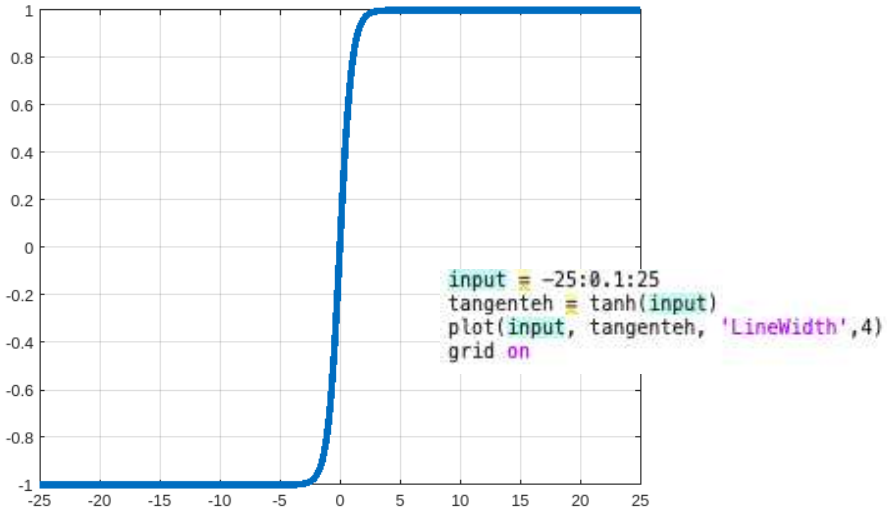
La función tangente hiperbólica se define como:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (3)$$

Devuelve valores entre -1 y 1. Es similar a la función sigmoide, pero tiene la ventaja de ser simétrica en torno al origen, lo que puede ser útil en algunos casos.

Figura 10

Función Tangente Hiperbólica.



Nota. Elaboración Propia en Matlab.

3.7.4 Función Softmax.

Se usa generalmente en la capa de salida de una red neuronal de clasificación múltiple, normaliza los valores de las neuronas de salida para que sumen 1, lo que los convierte en una distribución de probabilidad.

Gracias a la normalización del vector de salida de la red, la función softmax toma el valor con la probabilidad más alta y selecciona este valor como clasificación final del sistema.

Por ejemplo, el vector "n" definido a continuación pasa por la función softmax y se obtiene el vector "a".

n = [0 1 -0.5 0.5 0.25]

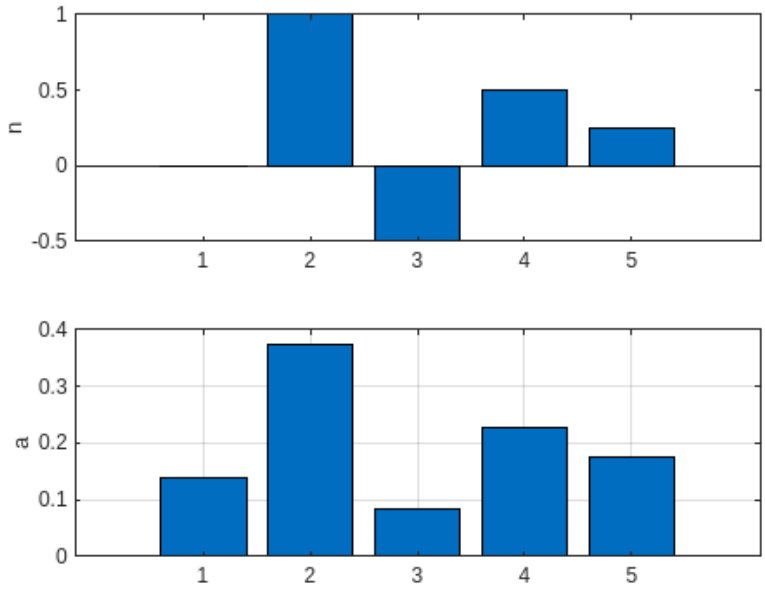
a = [0.1378 0.3745 0.0836 0.2272 0.1769]

La suma de los valores de "a" es igual a 1 y se toma la segunda posición del vector como la clasificación de la red porque tiene la mayor probabilidad.

Los resultados se muestran de forma gráfica en la figura 11.

Figura 11

Función Softmax.




```
n = [0; 1; -0.5; 0.5; 0.25];
a = softmax(n);
subplot(2,1,1), bar(n), ylabel('n')
subplot(2,1,2), bar(a), ylabel('a')
grid on
```

Nota. Elaboración Propia en Matlab.

En resumen, las funciones de activación son un componente fundamental en la construcción de redes neuronales. La elección de la función de activación apropiada puede afectar significativamente el rendimiento de la red en diferentes tipos de tareas.

3.8 Función de Coste o Función de Pérdida.

La función de coste, o función de pérdida, es una medida que sirve para evaluar qué tan bien la red neuronal está aprendiendo a realizar una tarea específica. Esta función calcula la diferencia entre la salida de la red neuronal y la salida esperada para el conjunto de datos de entrenamiento.

El objetivo del algoritmo de aprendizaje de la red neuronal durante el entrenamiento, es minimizar la función de coste, ajustando los parámetros de las neuronas para obtener la salida más cercana a la salida deseada.

El proceso de ajuste de pesos y parámetros de la red neuronal se realiza utilizando técnicas de optimización o algoritmos de aprendizaje descritas en el siguiente punto.

Existen diferentes tipos de funciones de error que se utilizan en función de la tarea que se desea realizar con la red neuronal. Algunos ejemplos comunes incluyen:

3.8.1 Error cuadrático medio (ECM).

Es mucho más usado en problemas de regresión, mide la diferencia entre la salida de la red y el valor real calculando el promedio de los errores elevados al cuadrado. El ECM está definido por la siguiente fórmula.

$$f(x) = \frac{1}{n} + \sum_{n=1}^{n} \left(\hat{Y}_i - Y_i\right)^2 \qquad (4)$$

Donde:

$\hat{Y}_i$: Es el valor de la predicción.

Y_i : Es valor real esperado.

n : número total de datos.

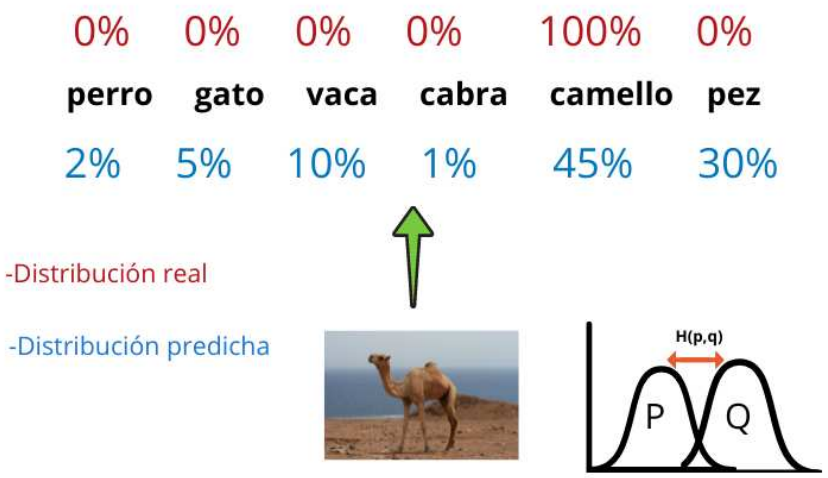
3.8.2 Entropía cruzada (cross-entropy).

La entropía cruzada mide la diferencia o distancia entre dos distribuciones de probabilidad de un conjunto de eventos, dando lugar a una función de perdida para problemas de clasificación.

Por ejemplo, para una red neuronal convolucional que clasifica una imagen de un animal, donde se tiene una función softmax en las neuronas de salida de la red que permite indicar las probabilidades que tiene la foto de pertenecer a cada una de las categorías, se obtiene el siguiente resultado.

Figura 12

Entropía Cruzada.



Nota. Adoptada de *Entropía de Shannon*, Rubén Cañadas, 2021, https://abdatum.com/ciencia/entropia-shannon.

En el resultado se toma la clase que tiene la probabilidad más alta y se muestra como la clasificación de la red.

La distribución de probabilidad generada por la red Neuronal, en color celeste, es la distribución modelada “q” y los resultados reales, de color rojo en la imagen, forman la distribución real “p”.

Así se va construyendo la función de pérdida de entropía cruzada y minimizándola en cada iteración del entrenamiento gracias al algoritmo de optimización, se irá reduciendo la diferencia entre ambas distribuciones y por lo tanto el modelo podrá generar resultados cada vez más reales.

Las funciones de coste están disponibles en la biblioteca TensorFlow de Python, elegir la función de coste adecuada para una tarea específica es importante para garantizar que la red neuronal pueda aprender y realizar la tarea de manera eficiente.

3.9 Algoritmo de Aprendizaje.

El algoritmo de aprendizaje de una red neuronal, también llamado optimizador, es el proceso que se utiliza para ajustar los pesos de las conexiones entre las neuronas durante el entrenamiento, de manera que la red pueda aprender a realizar una tarea específica.

Para redes convolucionales el algoritmo de aprendizaje ajusta los valores de los filtros definidos.

Existen diferentes algoritmos de aprendizaje que se utilizan en redes neuronales, y la elección del algoritmo depende de la tarea específica que se desea realizar y de las características de los datos de entrada, es importante utilizar el algoritmo adecuado para garantizar que el modelo pueda aprender de manera efectiva y eficiente. Algunos de los algoritmos de aprendizaje más comunes son:

3.9.1 Descenso de Gradiente.

La derivada en un punto de una función representa la pendiente en ese punto, el descenso de gradiente se basa en utilizar esa pendiente para ir moviéndose de manera descendente por la función de coste hasta lograr llegar a un punto mínimo.

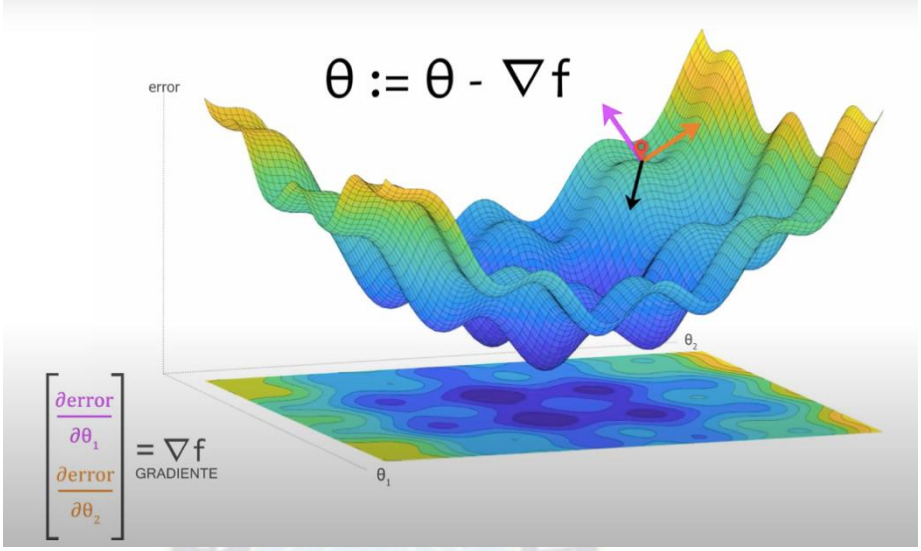
Para poder minimizar la función de coste se hace uso del vector gradiente, el cual se obtiene derivando la función de pérdida en derivadas parciales del error respecto a cada parámetro, usando este vector se puede actualizar los parámetros de la red para ir actualizando los mismos e ir optimizando la función.

En la figura 13 se muestra el grafico de una función de coste donde el error está representado en el eje “y” los parámetros θ_1 y θ_2 en los ejes “x” y “z” respectivamente.

Realizando las derivadas parciales, en el punto de color rojo, se obtiene el vector gradiente, de color negro, al cual se agrega el signo negativo para representar la dirección de descenso y así utilizar dicho vector para actualizar el vector de parámetros θ.

Figura 13

Descenso de gradiente.



Nota. Adaptado de *¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial*, Dot CSV, 2019, YouTube: https://www.youtube.com/watch?v=A6FiCDoz8_4

La ecuación final del descenso de gradiente es la siguiente:

$$\theta := \theta - \alpha * \nabla f \qquad (5)$$

Donde θ es el vector de parámetros y α es el ratio de aprendizaje o también llamado taza de aprendizaje, que controla el tamaño de los pasos de actualización, es decir que para valores grandes de α la actualización de los parámetros se dará de manera muy grande dando lugar a la no convergencia del algoritmo de optimización y para valores pequeños de alfa las actualizaciones se realizaran demasiado lentas, aumentando de manera considerable el tiempo de entrenamiento de la red.

El proceso se realiza de manera iterativa hasta llegar a un punto donde no se presente una variación notable del coste, es decir la pendiente sea casi nula, lo cual significa que se llegó a un mínimo local de la función.

Escoger el valor correcto del ratio de aprendizaje es fundamental para que el algoritmo de optimización funcione correctamente.

3.9.2 RMSprop (Root Mean Squared Propagation).

Es una variante del descenso de gradiente visto anteriormente, funciona calculando el gradiente de la función de pérdida con respecto a los parámetros, utilizándolo para minimizar la función de coste, pero la diferencia está en que RMSProp utiliza una media móvil de gradientes cuadrados para normalizar el gradiente. Esta normalización equilibra el tamaño del paso de actualización disminuyendo el paso para los gradientes grandes para evitar oscilación y aumentando el paso para los gradientes pequeños para evitar que tarde mucho el entrenamiento.

El algoritmo está definido por las siguientes fórmulas.

$$V_t = \beta * V_{(t-1)} + (1 - \beta) * dw^2 \quad (6)$$

$$\theta = \theta - \alpha * \frac{dw}{\sqrt{Vt} * \varepsilon} \quad (7)$$

Donde:

V_t : media móvil de los gradientes cuadrados.

β : hiperparámetro que controla la tasa de decaimiento de la media móvil.

θ: parámetros.

α : hiperparámetro que controla el tamaño del paso de la actualización.

dw : gradiente de la función de pérdida.

ε : constante épsilon para evitar la división por cero.

3.9.3 Adam.

Es un algoritmo de optimización estocástica eficiente que requiere gradientes de primer orden. El método calcula tasas de aprendizaje adaptativo individuales para diferentes parámetros a partir de estimaciones del primer y segundo momento de los gradientes.

El algoritmo Adam incorpora la ventaja del algoritmo de Descenso de Gradiente con Momentum, usa la media móvil para evitar oscilaciones excesivas en los valores de los

gradientes durante el entrenamiento, así también usa la ventaja de RMSprop que incorpora un tamaño de paso variable, que permite acelerar o desacelerar el proceso de entrenamiento dependiendo de la magnitud del gradiente en cada iteración.

Las dos ventajas mencionadas anteriormente, Adam las usa en la ecuación de actualización de los coeficientes. En particular, en cada iteración la actualización contiene un término asociado al momentum y otro que controla el tamaño del paso.

El pseudocódigo del algoritmo se presenta a continuación:

Definimos:

α : ratio de aprendizaje o tamaño del paso.

$\beta_1,\ \beta_2\ \in [0,1]$: Tasas de decaimiento exponencial para las estimaciones del momento.

$f(\theta)$: Función de coste estocástica con parámetros θ.

θ_0 : Vector de parámetros inicial.

$m_0\ \leftarrow\ 0$ (Inicializar vector de momento de primer orden)

$v_0\ \leftarrow\ 0$ (Inicializar vector de momento de segundo orden)

$t\ \leftarrow\ 0$ (Inicializar timestep)

$while\ (\theta_t\ no\ converge\ )\ do$

$t \leftarrow t + 1$

$g_t\ \leftarrow\ \nabla_t f_t(\theta_{t-1})$: $Obtener\ gradientes\ con\ respecto\ al\ coste\ estocástico\ en\ el\ paso\ de\ tiempo\ t.$

$m_t\ \leftarrow\ \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$: $Actualizar\ estimación\ sesgada\ del\ primer\ momento.$

$v_t\ \leftarrow\ \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$: $Actualizar\ la\ estimación\ sesgada\ del\ segundo\ momento.$

$\widehat{m_t}\ \leftarrow \dfrac{m_t}{(1-\beta_1^t)}$: $Calcular\ la\ estimación\ del\ primer\ momento\ con\ corrección\ de\ sesgo.$

$\widehat{v_t}\ \leftarrow \dfrac{v_t}{(1-\beta_2^t)}$: $Calcular\ la\ segunda\ estimación\ del\ momento\ corregida\ por\ el\ sesgo$

$\theta_t\ \leftarrow\ \theta_{t-1}\ -\ \alpha * \dfrac{\widehat{m_t}}{\left(\sqrt{\widehat{v_t}}\ +\ \varepsilon\right)}$: $Actualización\ de\ parámetros.$

$end\ while$

$return\ \theta_t$

El ajuste mencionado de los pesos de las conexiones de la red, puede realizarse a grandes o pequeños pasos, eso depende de uno o muchos hiperparámetros con los que cuenta cada algoritmo de aprendizaje, por ejemplo, para el algoritmo Adam se define:

α = 0.001

β_1= 0.9

β_2= 0.999

El pseudocódigo y los valores definidos de los hiperparámetros se encuentra en el artículo: “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION” de Diederik P. Kingma y Jimmy Lei Ba, publicado en 2015.

Los optimizadores mencionados se encuentran en la biblioteca TensorFlow de Python y pueden usados variando sus hiperparámetros o manteniendo los definidos por defecto.

El valor del hiperparámetro puede afectar en la convergencia del aprendizaje de la red, es por eso que se debe escoger un valor adecuado para obtener mejores resultados en menores tiempos de entrenamiento.

3.10 Dropout.

El Dropout en una red neuronal consiste en desconectar neuronas en el proceso de entrenamiento de la misma, esta eliminación o desconexión se realiza de manera aleatoria y da lugar a evitar o disminuir el sobre entrenamiento o sobreajuste de la red.

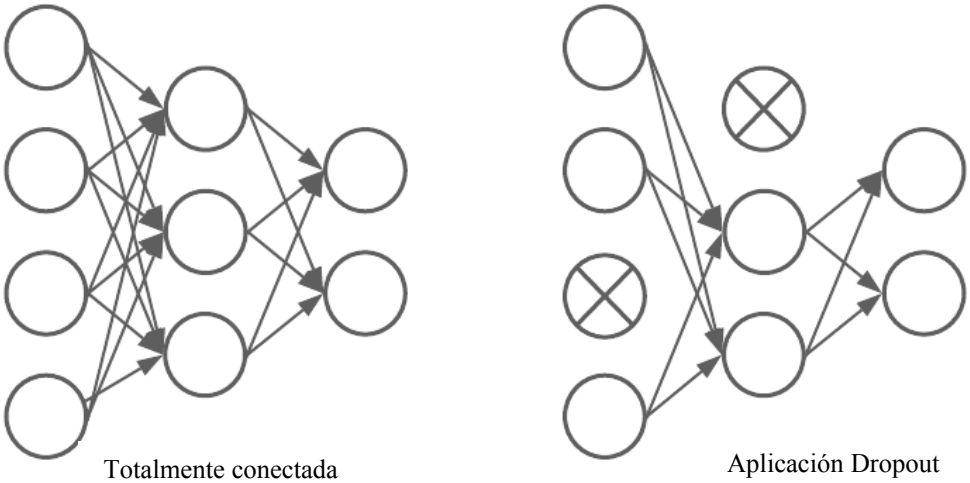
El sobre entrenamiento de una red se presenta cuando una red neuronal aprende de memoria los datos de entrenamiento, pero al momento de ser validado con datos que no han sido utilizados en el entrenamiento, es decir datos que nunca vio, la red genera malos resultados.

Es por eso que es necesario evitar el sobre entrenamiento con diferentes procesos como por ejemplo el pre procesamiento de la información, aumento de los datos de entrenamiento o aplicando Dropout.

La imagen de la figura 14 muestra la aplicación de Dropout en una red neuronal del tipo densa, como se puede observar en la imagen de la izquierda se muestra la red totalmente conectada y en la derecha la misma red con algunas neuronas eliminadas.

Figura 14

Demostración de Dropout.



Nota. Elaboración propia en Lucid.app.

Con menor cantidad de neuronas se evita el sobreajuste de la red, en la optimización del sistema la desconexión de neuronas se va ajustando de manera que la red mejore en su eficiencia y en la minimización de la función de coste.

3.11 Backpropagation.

Backpropagation o retropropagación es el algoritmo que permite que la red neuronal aprenda de forma independiente a ajustar sus parámetros en el entrenamiento, gracias a la retropropagación se puede ahorrar un gran costo computacional y disminuir considerablemente el tiempo del proceso de optimización de la red.

La mayoría de los algoritmos de aprendizaje se basan en el cálculo del gradiente de una función de coste para poder minimizarla, pero en una red grande conformada por muchas neuronas es muy difícil realizar dicho cálculo de forma convencional, ya que para obtener el vector gradiente se debe calcular las derivadas parciales de todos los parámetros de la red.

El algoritmo de retropropagación comienza en la última capa, la capa de salida de la red, en la cual se responsabiliza a cada neurona con el error obtenido, se observa qué neurona ha tenido más culpa en generar el error entonces a esa neurona se le da mayor responsabilidad, esto es

representado por el error imputado a la neurona, luego se pasa a una capa anterior y se repite el proceso, el error es propagado hacia atrás verificando qué neurona tiene más responsabilidad del error final generado, así se continúa el camino hacia atrás de la red capa por capa.

De esta forma el algoritmo de retropropagación permite calcular el gradiente de la función de coste capa por capa y no neurona por neurona.

Para calcular el gradiente se tiene que desarrollar la siguiente derivada, ¿Cómo varía el coste ante un cambio de los parámetros "w" y "b"?

$$\frac{\partial C}{\partial w}, \qquad \frac{\partial C}{\partial b} \tag{8}$$

Tomando como ejemplo la red neuronal de la figura 15 se puede observar que para obtener la salida de la red, los parámetros de una capa anterior pasan por las siguientes funciones:

$C(x)$: Función de coste.

$a(x)$: Función de activación.

$Z^L = W^L * a^{L-1} + b^L$: Suma ponderada en las neuronas.

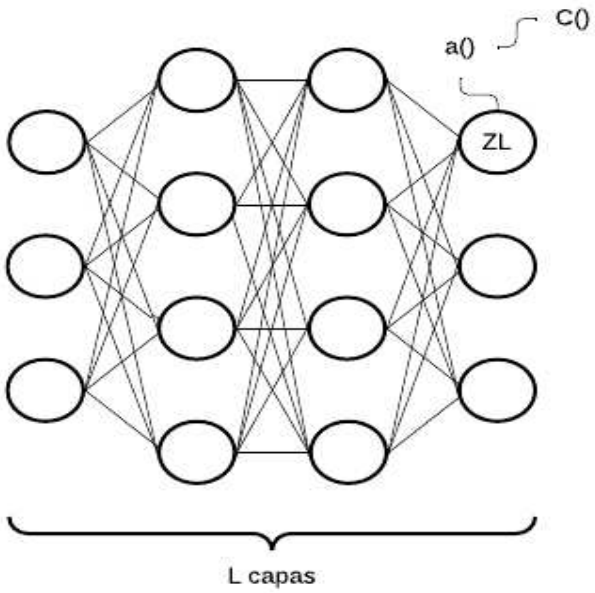
Obteniendo, en la última capa de la red, la siguiente composición de funciones:

$$C(a(Z^L)) = error \tag{9}$$

En Z^L los pesos de la última capa W^L están conectados con la salida de la capa anterior, dicha salida está representada por la salida de la función de activación de la capa anterior a^{L-1}.

Figura 15

Red Neuronal ejemplo



Nota. Elaboración propia Lucid.app.

Entonces para obtener las derivadas necesarias (8) se hace uso de la regla de la cadena para derivar la composición de funciones, obteniendo:

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial a^L} * \frac{\partial a^L}{\partial z^L} * \frac{\partial z^L}{\partial w^L} \quad (10)$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} * \frac{\partial a^L}{\partial z^L} * \frac{\partial z^L}{\partial b^L} \quad (11)$$

En donde, para la ecuación (12):

$$z^L = \sum_i a_i^{L-1} w_i^L + b^L \quad (12)$$

Las derivadas son:

$$\frac{\partial z^L}{\partial w^L} = a_i^{L-1} \quad (13)$$

$$\frac{\partial z^L}{\partial b^L} = 1 \quad (14)$$

El término común en las dos ecuaciones (10) y (11) representan al error imputado a la neurona δ^L el cual se utiliza para responsabilizar a las neuronas, es decir si el valor de δ^L es grande significa que un pequeño cambio en los parámetros de esa neurona se ve reflejado directamente en el error, mientras que si el valor es pequeño significa que cambiando sus parámetros no afectará mucho al error.

$$\delta^L = \frac{\partial C}{\partial a^L} * \frac{\partial a^L}{\partial z^L} \quad (15)$$

Una vez calculado el error en la última capa, se pasa a una capa anterior retropropagando el error calculado en la última capa, esto se logra multiplicando δ^L con la matriz de parámetros W^L que conecta ambas capas, distribuyendo el error por las neuronas. Todo este proceso es extensible a las demás capas de la red.

Retropropagación del error a una capa anterior.

$$\delta^{l-1} = W^l * \delta^l * \frac{\partial a^{l-1}}{\partial z^{l-1}} \quad (16)$$

En (16) el término $\frac{\partial a^{l-1}}{\partial z^{l-1}}$ representa a la derivada de la función de activación.

Antes de pasar a una capa anterior se realiza el cálculo de las derivadas parciales de los parámetros de toda la capa en la que se encuentra.

Cálculo de las derivadas de la capa usando el error

$$\frac{\partial C}{\partial b^{l-1}} = \delta^{l-1} \quad (17)$$

$$\frac{\partial C}{\partial w^{l-1}} = \delta^{l-1} * a^{l-2} \quad (18)$$

De esta forma se obtiene todas las derivadas necesarias para formar el vector gradiente, se realiza el cálculo de las expresiones (15), (16), (17) y (18) en cada capa de la red para tener el error en la capa actual con respecto a la capa anterior ya calculado, recorriendo de adelante hacia atrás, reduciendo significativamente el cálculo computacional.

SECCION III. INGENIERIA DEL PROYECTO.

4 Capítulo 4 – Análisis de Requerimientos.

4.1 Requerimientos.

El sistema tiene que ser capaz de identificar y calificar si la imagen de rayos X del tórax del paciente, presenta neumonía viral, neumonía bacteriana o si no presenta ningún tipo de neumonía.

Al tratar con imágenes es imprescindible diseñar una red neuronal del tipo convolucional, ya que una red neuronal tipo densa no es suficiente para obtener buenos resultados.

El set de datos de imágenes de radiografías de tórax consta de 3600 imágenes de diferentes tamaños, las cuales deben ser adecuadas para utilizarlas en el entrenamiento.

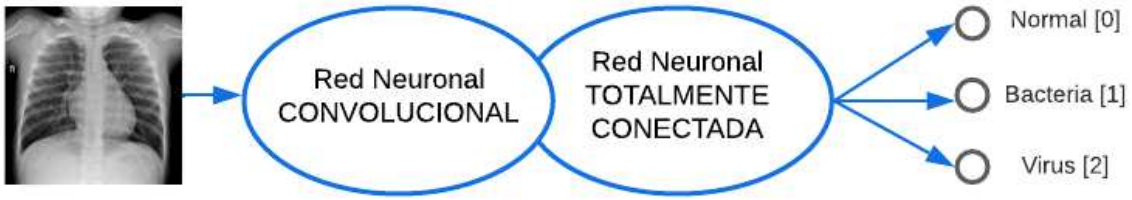
4.2 Esquema General.

Al trabajar con imágenes es necesario la implementación de una red neuronal convolucional, la cual, para este caso, presenta una mayor ventaja sobre una red neuronal del tipo densa.

La red está representada por el siguiente esquema.

Figura 16

Esquema general de la RNC.



Nota. Elaboración propia en lucid.app.

Se puede notar las dos partes de la red, la primera etapa convolucional para obtener características de la imagen y la segunda etapa de la red tipo densa para realizar la clasificación en base a la información obtenida de la anterior etapa.

La entrada a la red es una imagen y la salida de la red es un vector, la capa de salida tiene que contener tres neuronas ya que solo se tiene tres clases para clasificar las imágenes.

4.3 Funcionalidades.

Gracias a la arquitectura implementada el sistema es capaz de realizar la clasificación de imágenes de radiografía de tórax de la siguiente forma:

Figura 17

Clasificación de radiografías de tórax.

Nota. Adoptado de *Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning*, Kermany D., Goldbaum M., Cai W., Lewis M., & Kang Zhang X., 22 de febrero de 2018. De CellPress Sitio web: https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5#relatedArticles.

La radiografía de tórax normal, imagen de la izquierda, muestra pulmones claros sin áreas de opacificación anormal en la imagen, lo que significa que el paciente no presenta ningún tipo de neumonía.

La neumonía bacteriana, imagen del medio, normalmente muestra una consolidación lobar focal, en este caso en el lóbulo superior izquierdo, indicado con flechas blancas.

Mientras que la neumonía viral, derecha, se manifiesta con un patrón "intersticial" (que se encuentra entre los espacios pequeños) más difuso en ambos pulmones.

La red neuronal va extrayendo de la imagen de radiografía de tórax las características explicadas anteriormente gracias a la parte convolucional, luego la parte densa de la red se encargada de la clasificación en una de las tres clases dependiendo de la imagen.

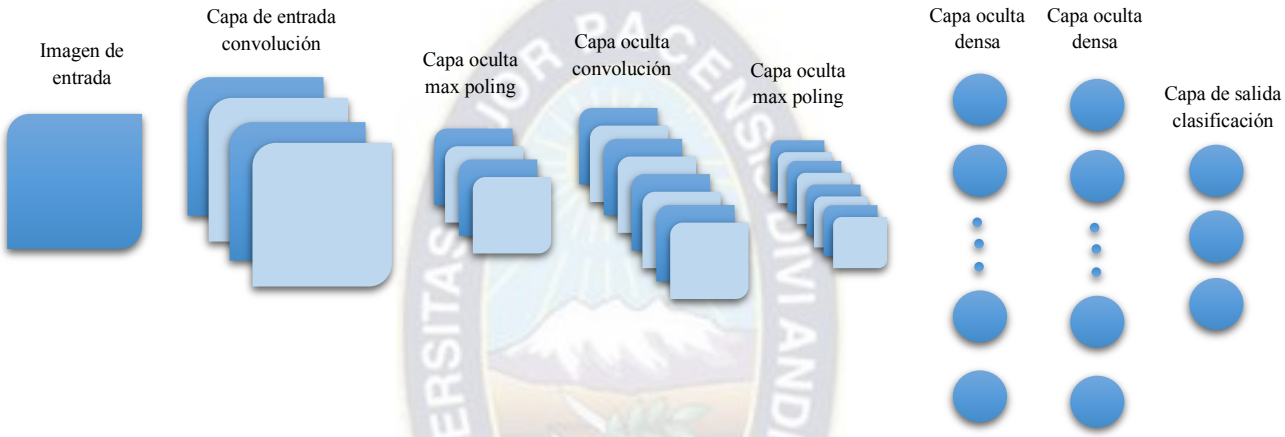
5 Capítulo 5 – Diseño del Sistema.

5.1 Arquitectura General.

La red neuronal convolucional presenta la siguiente arquitectura general:

Figura 18

Arquitectura de la RNC.



Nota. Elaboración propia.

La imagen de entrada es preprocesada y se obtiene una imagen en escala de grises de tamaño 225x225 pixeles y un solo canal de color.

Se puede diferenciar en la figura que existen dos tipos de redes, la primera parte de la red neuronal consiste en una red neuronal convolucional, la cual tiene como objetivo extraer características específicas de la imagen, la segunda parte de la red consiste en una red neural densa.

La red convolucional comienza con una capa que consta de 64 filtros, esta capa extrae las características más generales de la imagen, como por ejemplo curvas o bordes, luego continua por una capa de max poling para la reducción de tamaño de la imagen, a continuación sigue una capa de convolución de 128 filtros, lo que permite ahora es extraer características más objetivas como segmentación de partes que contienen características similares y así sucesivamente hasta

poder llegar a una segmentación de las partes importantes que muestran una posible infección del pulmón.

A continuación, se muestra las características de cada capa convolucional en la siguiente tabla.

Tabla 1

Capas de la etapa de la red Convolucional.

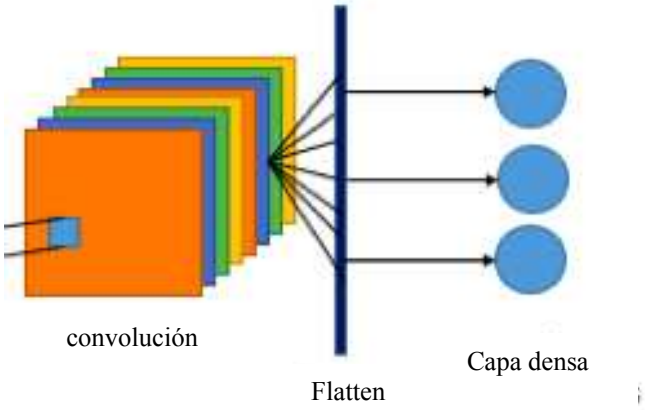
Nro. Capa	Nro. Filtros	Pooling	Func. Activación
1	64	Max Pooling	Relu
2	128	Max Pooling	Relu
3	256	Max Pooling	Relu
4	512	Max Pooling	Relu

Saliendo de la última capa de convolución, la información entra a la capa nro. 5, esta capa es del tipo Flatten, la cual realiza un aplanamiento de la imagen que está siendo procesada en las anteriores capas de convolución, al pasar por esta capa la imagen de 3 dimensiones se transforma en una sola dimensión para poder entrar a la etapa de la red neuronal tipo densa.

En la figura 18, se muestra un ejemplo de aplicación de una capa Flatten en una etapa de una red neuronal.

Figura 19

Demostración Flatten.



Nota. Adaptado de *Deep learning aplicado para la detección de hemorragias y tumores cerebrales*, Mauricio Fernando Hidalgo Barrientos, Bryan Isaac Hayes Ortiz, Ignacio Delgadillo Vera, Manuel Goyo Escalona, 27 de octubre de 2021, Artículo: https://brapci.inf.br/index.php/res/download/169547.

Como se puede ver la capa Flatten sirve para unir una red neuronal convolucional con una red neuronal tipo densa.

Gracias a la red neuronal convolucional, la red del tipo densa realiza la clasificación tomando en cuenta las características de la imagen de entrada extraídas en la primera etapa.

Una red neuronal densa por si sola, realiza la clasificación de la imagen tomando en cuenta solo la posición de los pixeles, no le importa las características de la imagen, dando lugar a un sistema erróneo ya que inclinando ligeramente la imagen de rayos X de tórax afecta totalmente en la clasificación dando malos resultados.

El esquema de la red neuronal densa se muestra en la tabla número 2.

Tabla 2

Capas de la etapa de la red Densa.

Nro. Capa	Nro. Neuronas	Func. Activación	
5	Flatten		
6	256	Relu	Dropout
7	128	Relu	Dropout
8	64	Relu	Dropout
9	3	Softmax	

Las capas 6, 7 y 8 son neuronas totalmente conectadas con función de activación Relu la cual introduce no linealidad al sistema para poder obtener mejores resultados en el entrenamiento de la red.

Además, se hace uso de Dropout en las capas densas para evitar el sobreajuste de la red en el entrenamiento desconectando aleatoriamente en cada época de entrenamiento diferentes neuronas.

La capa número 9 es la capa de salida de la red, consta de tres neuronas, con función de activación softmax para que la suma de las probabilidades de las tres neuronas sea igual a 1, de esta forma la neurona con probabilidad más alta representará el resultado de la clasificación de la imagen.

5.2 Pre procesamiento de las Imágenes y Aumento de Datos.

El dataset para el entrenamiento consiste en imágenes de radiografías de tórax de pacientes, cada una clasificada en las siguientes clases:

Radiografías de tórax normal, 1200 imágenes.

Radiografías de tórax que presentan Neumonía Viral, 1200 imágenes.

Radiografías de tórax que presentan Neumonía Bacteriana, 1200 imágenes.

Antes de comenzar el entrenamiento el dataset completo pasa por un pre procesamiento de la siguiente forma:

- Todas las imágenes se redimensionan a un solo tamaño: 224x224x3.
- Se convierten a escala de grises: 224x224x1.
- Normalización de los pixeles de valores de 0 a 255 a valores de 0 a 1.

Después del preprocesamiento de las imágenes se realiza un aumento de datos para mejorar el entrenamiento y reducir el sobreajuste de la siguiente manera:

- Rotación aleatoria de 23 grados.
- Desplazamiento de la imagen hacia arriba o abajo de un 10%.
- Desplazamiento de la imagen hacia derecha o izquierda de un 10%.
- Acercamiento de 110% y alejamiento de 80%.

A continuación, en la figura 20, se muestra la comparación de las imágenes después del aumento de datos, a la izquierda las imágenes originales y a la derecha se puede notar las mismas imágenes con pequeñas alteraciones para el aumento de datos, lo cual ayuda en un mejor entrenamiento ya que las imágenes de radiografías de radio X de tórax siempre presentan pequeñas alteraciones debido al movimiento de los pacientes en la toma de la radiografía.

Figura 20

Aumento de datos.

Nota. Comparación de imágenes radio X de tórax antes y después del aumento de datos. Elaboración propia en python.

Luego del aumento de datos se realiza la separación de un 15% de imágenes del dataset, para realizar la validación del entrenamiento a medida que va avanzando.

5.3 Programación.

El tipo de programación a implementar es del tipo estructurada siguiendo el diagrama de flujo mostrado en la figura 21.

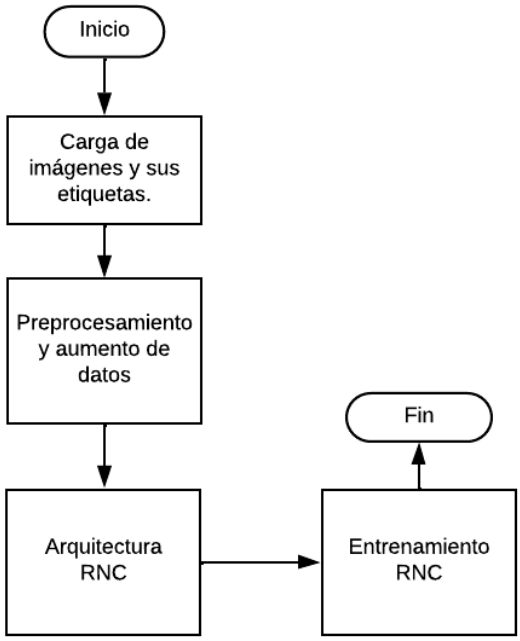
La carga de imágenes es un bucle donde se carga una imagen a la vez con su respectiva etiqueta, el preprocesamiento se realiza de igual forma, para cada imagen en un bucle, mientras que el aumento de datos se realiza por lotes de 32 imágenes.

Luego se define la arquitectura de la red neuronal convolucional junto al optimizador y función de coste que se va a utilizar para el entrenamiento.

Finalmente se realiza el entrenamiento y se obtiene el modelo de la RNC final.

Figura 21

Diagrama de Flujo de Programación.



Nota. Elaboración propia en lucid.app.

6 Capítulo 6 – Entrenamiento de la Red Neuronal.

6.1 Entrenamiento.

Gracias al dataset obtenido el entrenamiento de la red se realizó de manera supervisada, es decir que cada imagen viene con su clasificación correcta lo que permite entrenar y ajustar los parámetros de la red observando, comparando y optimizando la predicción de la red con la clasificación real de la imagen, para realizar este proceso se utilizó:

- Función de coste: Entropía cruzada (cross-entropy).
- Optimizador: Adam.

El entrenamiento se realizó en lotes de tamaño 32 en un total de 160 etapas, variando diferentes tipos de configuraciones de redes neuronales, probando diferentes funciones de activación, aumentando y disminuyendo capas y así sucesivamente hasta obtener el mejor resultado de una red neuronal demostrado en el capítulo 7 del presente documento.

El tiempo de entrenamiento fue variando en cada prueba de distintas redes, para la red neuronal final se obtuvo un tiempo de entrenamiento de 32 minutos. La máquina virtual donde se realizó el entrenamiento tiene las siguientes características:

- GPU T4.
- RAM de GPU de 15gb (gigabytes) de los cuales se usó 8.5gb para el entrenamiento.
- RAM de sistema de 51gb de los cuales se usó solamente 10.4gb en el entrenamiento.
- Disco de 166gb de memoria del cual se usó 27.2gb para el entrenamiento.

Se puede realizar una estimación del número total de parámetros optimizados en el entrenamiento de la siguiente manera:

Para la etapa densa se tiene en la capa 6 de la red 256 neuronas conectadas con las 128 neuronas de la siguiente capa, entonces en dicha capa se obtiene 32768 parámetros, más los parámetros de bias (256) se tiene 33024 de los cuales el 80% son desconectados gracias al dropout de esta capa, dando un total de 6604 parámetros.

Siguiendo el mismo cálculo se tiene para la capa 7 un total de 4160 parámetros y para la capa 8 179 parámetros.

Para las primeras capas de la red neuronal convolucional, los parámetros que se tienen que ajustar corresponden a la cantidad de filtros que tiene cada capa, es decir. 64 parámetros para la primera capa, 128 para la segunda, 256 para la tercera y 512 para la cuarta capa.

Finalmente sumando todos los parámetros de todas las capas se obtiene 11903 parámetros en la red neuronal convolucional.

El número total de parámetros calculado solo es una estimación ya que el dropout se aplica de manera aleatoria en el entrenamiento además que no se tomaron en cuenta los términos de bias en las primeras capas de la red.

SECCION IV. ETAPA CONCLUSIVA.

SECCION IV. ETAPA CONCLUSIVA.

7 Capítulo 7 – Pruebas y Resultados.

7.1 Pruebas

Se probaron diferentes características sobre la arquitectura general diseñada de la red neuronal convolucional, observando las ventajas y desventajas de cada una hasta obtener la mejor red.

Para todos los gráficos se tiene la siguiente representación:

- Curva con datos de entrenamiento color Azul.
- Curva con datos de validación color Celeste.

Probando en primer lugar filtros de tamaño 4x4, una configuración de la etapa densa de la red de 50, 150, 50, 3, y sin dropout, se obtienen la primera red neuronal convolucional.

Tabla 3

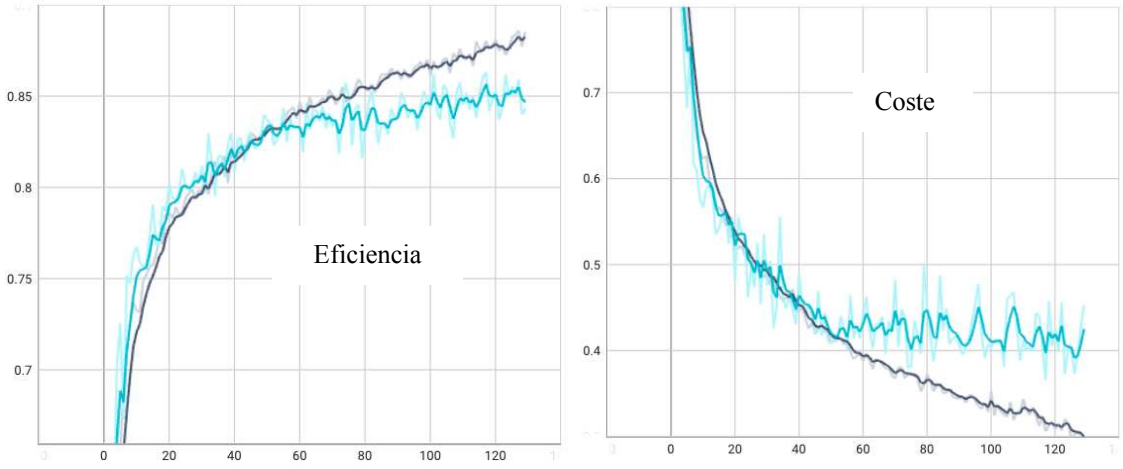
Red Convolucional 1.

Nro. Capa	Nro. Filtros	Nro. Neuronas	Max Poling	Func. Activación	Dropout
1	64 (4x4)		2x2	Relu	
2	128 (4x4)		2x2	Relu	
3	256 (4x4)		2x2	Relu	
4	512 (4x4)		2x2	Relu	
5	512 (4x4)			Relu	
6			Flatten		
7		50		Relu	
8		150		Relu	
9		50		Relu	
10		3		Softmax	

Gráficos de los resultados de esta red convolucional 1 se muestran en la figura 20.

Figura 21

Eficiencia y costo RNC1



Nota. RNC1. Elaboración propia en python.

Al ver que las curvas se separan tanto en la comparación de la eficiencia como en la función de coste se puede llegar a la conclusión que la red presenta un sobreajuste, es decir, la red funciona mucho mejor para datos de entrenamiento, pero al ser testeada con datos de validación la red ya no puede funcionar de la misma manera obteniendo malos resultados.

El sobreajuste obtenido significa que la red aprende de memoria los datos de entrenamiento y por ende genera malos resultados a la hora de realizar la validación con datos que jamás han sido vistos por la red,

Reduciendo el tamaño de los filtros a 3x3 y cambiando la configuración de la etapa densa de la red a 256, 128, 64, 3, se obtiene la red convolucional número 2.

Tabla 4

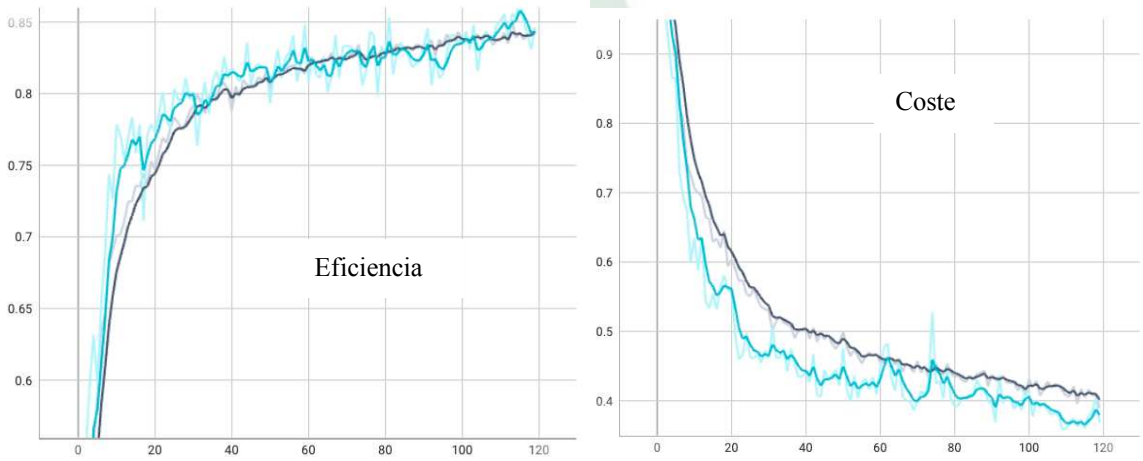
Red Convolucional 2.

Nro. Capa	Nro. Filtros	Nro. Neuronas	Max Poling	Func. Activación	Dropout
1	64 (3x3)		2x2	Relu	
2	128 (3x3)		2x2	Relu	
3	256 (3x3)		2x2	Relu	
4	512 (3x3)		2x2	Relu	
5			Flatten		
6		256		Relu	0.8
7		128		Relu	0.5
8		64		Relu	0.3
9		3		Softmax	

Gráficos de los resultados de esta red:

Figura 22

Eficiencia y costo RNC2.



Nota. Elaboración propia en python.

La eficiencia de la red convolucional 2 es menos del 85%, gracias al Dropout se reduce el sobreajuste de la red y la función de coste baja hasta aproximadamente 0.4, resultados que fueron mejorados en la red neuronal final.

Manteniendo los filtros 3x3, modificando la etapa densa a 512, 256, 3 y aplicando Dropout se tiene los siguientes resultados.

Tabla 5

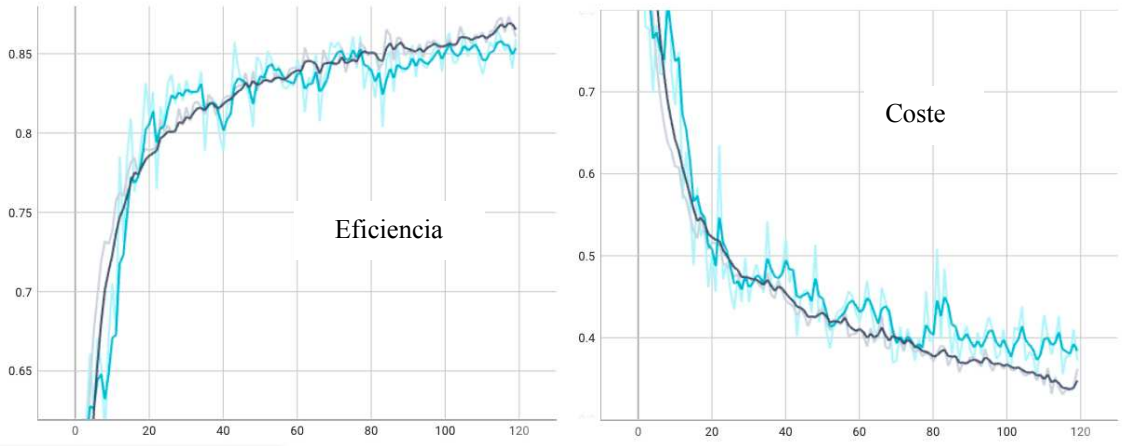
Red Convolucional 3.

Nro. Capa	Nro. Filtros	Nro. Neuronas	Max Poling	Func. Activación	Dropout
1	64 (3x3)		2x2	Relu	
2	128 (3x3)		2x2	Relu	
3	256 (3x3)		2x2	Relu	
4	512 (3x3)		2x2	Relu	
5			Flatten		
6		512		Relu	0.5
7		256		Relu	0.5
8		3		Softmax	

Gráficos de los resultados de esta red en la figura 22:

Figura 23

Eficiencia y costo RNC3.



Nota. Eficiencia y costo RNC3. Elaboración propia.

La eficiencia de la red sube un poco más del 85% pero se nota un pequeño sobreajuste de la red comparando las curvas de la función de coste.

Finalmente se obtiene el mejor arreglo de la red neuronal que permite generar mejores resultados, la red neuronal convolucional final y sus resultados se presenta en el siguiente punto

7.2 Resultados Finales.

Después de varias pruebas sobre una misma arquitectura de red, variando algunas características de la misma, se logró diseñar la red neuronal convolucional final para que trabaje en el prototipo del sistema propuesto en el presente proyecto.

En la tabla número 6 está desarrollada la red neuronal convolucional final, mostrando cada capa y las características de cada una.

Tabla 6

Red Convolucional Final.

Nro. Capa	Nro. Filtros	Nro. Neuronas	Max Poling	Func. Activación	Dropout
1	64 (4x4)		2x2	Relu	
2	128 (4x4)		2x2	Relu	
3	256 (4x4)		2x2	Relu	
4	512 (4x4)		2x2	Relu	
5			Flatten		
6		256		Relu	0.8
7		128		Relu	0.5
8		64		Relu	0.3
9		3		Softmax	

Como se observa la etapa convolucional consta de filtros de tamaño 4x4 además de Max Pooling después de cada convolución, toda la etapa de convolución tiene una función de activación Relu.

La etapa densa de la red consta de 5 capas, la primera es Flatten, luego siguen 3 capas ocultas con activación Relu, además de Dropout después de cada capa oculta y finalmente la capa de salida con activación Softmax.

Los resultados después de las 160 etapas de entrenamiento de la red neuronal convolucional final se muestran a continuación en la figura 23.

Figura 24

Últimas etapas de entrenamiento.

```
Epoch 156/160
97/97 [==============================] - 10s 107ms/step - loss: 0.2976 - accuracy: 0.8794
Epoch 157/160
97/97 [==============================] - 10s 106ms/step - loss: 0.3158 - accuracy: 0.8797
Epoch 158/160
97/97 [==============================] - 10s 107ms/step - loss: 0.3134 - accuracy: 0.8832
Epoch 159/160
97/97 [==============================] - 10s 107ms/step - loss: 0.3090 - accuracy: 0.8848
Epoch 160/160
97/97 [==============================] - 10s 106ms/step - loss: 0.3140 - accuracy: 0.8755
```

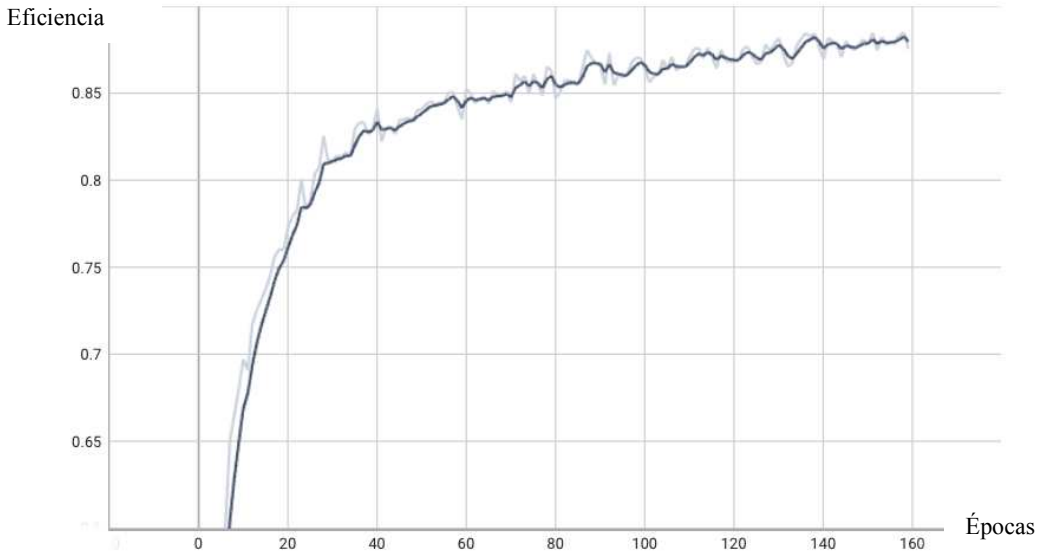
Nota. Python tensorflow. Elaboración propia.

Se puede notar que la red entrenada final tiene una eficiencia del 87.55% y el resultado de la evaluación de la función de coste de 0.3140.

Las curvas del aprendizaje de la red se muestran en las siguientes figuras, eficiencia y función de coste representadas a medida que van pasando las épocas de entrenamiento.

Figura 25

Eficiencia de red.

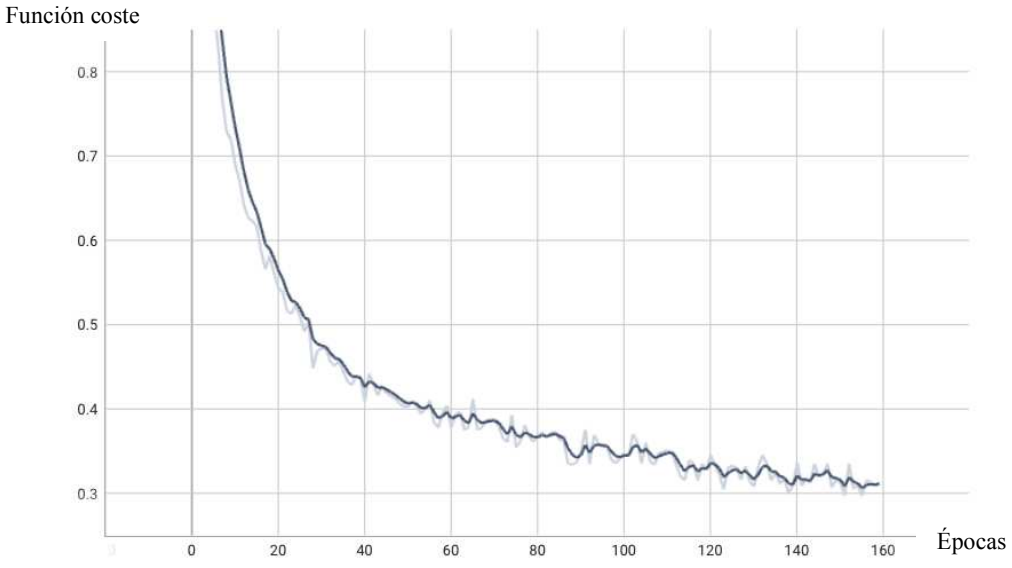


Nota. Elaboración propia en python tensorboard.

Se observa que en el tramo comprendido entre la época 140 y la 160, la eficiencia se va alterando en valores muy pequeños, es por eso que se escoge 160 etapas suficientes para entrenar la red.

Figura 26

Función de coste.



Nota. Elaboración propia en python tensorboard.

La minimización de la función de coste es notable en el gráfico, ya que se logra minimizar la perdida desde un valor mayor a 1 hasta aproximadamente un valor de 0.3.

Los resultados mostrados anteriormente están dados gracias al entrenamiento de la red neuronal convolucional únicamente con las imágenes de entrenamiento pasando primero por el preprocesamiento y luego por el aumento de datos, la validación de la red obtenida se muestra en el siguiente punto.

7.3 Validación.

Para la validación de la red neuronal convolucional se dividió el set de entrenamiento en 85% para el entrenamiento y el 15% de los datos para la validación, a continuación, se muestra una tabla comparativa en las últimas 4 etapas de entrenamiento.

Tabla 7

Comparación de resultados del entrenamiento con la validación.

Nro. Etapa	Coste Entrenamiento	Eficiencia Entrenamiento	Coste Validación	Eficiencia Validación
157	0.3158	87.97 %	0.3043	88.40 %
158	0.3134	88.32 %	0.3055	87.20 %
159	0.3090	88.48 %	0.3208	85.80 %
160	0.3140	87.55 %	0.2904	88.60 %

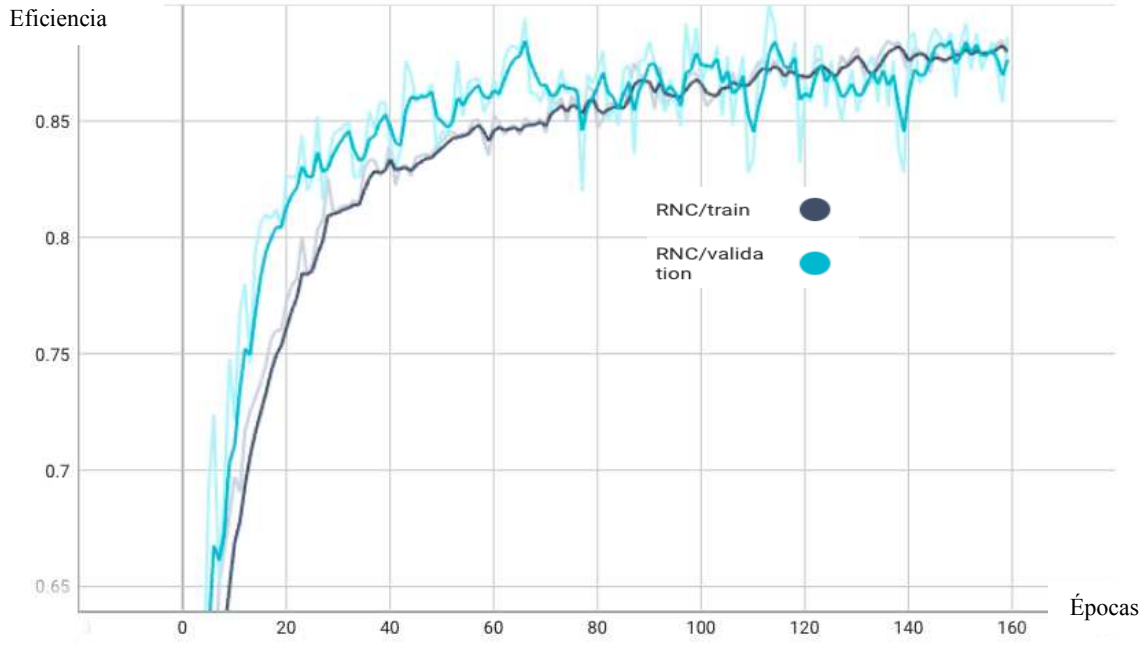
Como se muestra en la tabla, la red neuronal convolucional entrenada fue testeada con las imágenes de validación, es decir imágenes que nunca vio durante el entrenamiento, danto un resultado del 88.60% de eficiencia.

Además, que en la validación se obtiene un menor valor de la evaluación de la función de coste, 0.2904, lo que significa que la red a aprendido a generalizar el conocimiento sobre el cual ha sido entrenada.

En la siguiente figura se representa la comparación de las curvas de entrenamiento y validación del modelo de red neuronal convolucional en un gráfico de eficiencia vs épocas de entrenamiento.

Figura 27

Comparación entre entrenamiento y validación.



Nota. Python tensorboard, elaboración propia.

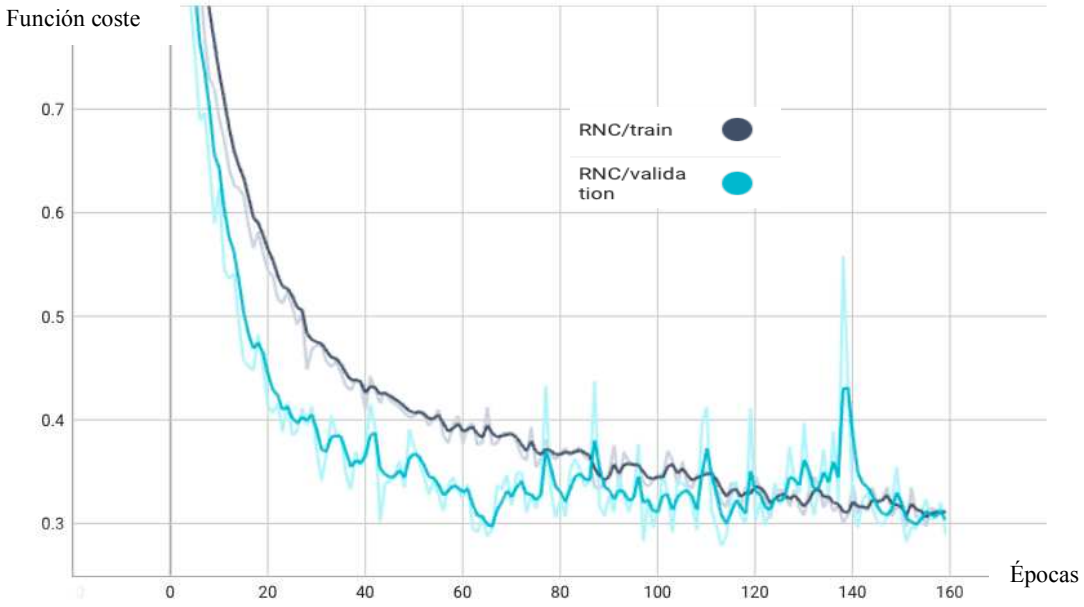
Se puede observar que el aprendizaje de la red va aumentando considerablemente en las primeras etapas de entrenamiento, avanzando hasta llegar a estabilizarse en las últimas etapas.

La curva de la eficiencia con los datos de validación, de color celeste, va creciendo junto a la curva de entrenamiento, de color azul, a pesar de presentar picos más inestables la curva de validación converge junto a la curva de entrenamiento observando las ultimas 10 etapas, esto demuestra que la red entrenada genera resultados aceptables frente a imágenes que no fueron utilizadas en su entrenamiento.

El siguiente gráfico es la representación de los resultados de las funciones de coste de entrenamiento y validación en cada época de entrenamiento

Figura 28

Función de coste entrenamiento y validación.



Nota. Python tensorboard, elaboración propia

La forma en la que convergen las dos curvas de las funciones de coste, celeste para validación y azul para entrenamiento, indican que no existe un sobre ajuste de la red neuronal, esto significa que la red neuronal está realizando una generalización del conocimiento y no está memorizando los datos de entrenamiento, dando así buenos resultados en sus predicciones.

8 Capítulo 8 – Conclusiones y Recomendaciones.

8.1 Conclusiones.

Se logró diseñar la red neuronal convolucional que permite detectar si un paciente presenta neumonía viral, neumonía bacteriana, o si no presenta ningún tipo de neumonía, partir de una imagen de radiografía de tórax, con una eficiencia del 87.55%, la cual puede servir de base para desarrollar un sistema completo de detección y diagnóstico de neumonía.

Gracias al algoritmo de Backpropagation, el Deep learning ha sido implementado sin complejidades dando lugar al ajuste de los aproximadamente 11903 parámetros de la red neuronal diseñada.

Se puede verificar que el preprocesamiento de las imágenes del dataset ayuda significativamente a mejorar el entrenamiento de la red, ya que este paso da lugar a reducir el costo computacional del procesamiento de las imágenes en la etapa del entrenamiento de la red neuronal.

También es bueno recalcar que el aumento de datos ayuda a obtener mejores resultados en la eficiencia de la red, ya que introduciendo pequeñas alteraciones a las imágenes del set de entrenamiento se consigue reducir el error de sesgo que se puede producir en la toma de radiografías de tórax de los pacientes, así se generaliza aún más el entrenamiento de la red.

Se logró desarrollar el prototipo de un sistema inteligente que generalizó el conocimiento, superando a un sistema experto que memoriza las imágenes y sus etiquetas en el entrenamiento.

8.2 Recomendaciones.

La mayor parte del proceso de diseño, entrenamiento, validación e implementación de la red neuronal artificial se realizó en máquinas virtuales en la nube de Google, las cuales se encargaron de todo el costo computacional requerido para todas las pruebas realizadas. Google ofrece, por el pago de una membresía, máquinas virtuales que cuentan con GPUs de Nvidia como A100, V100 y T4 y memorias RAM amplias de hasta 51gb de tamaño, por lo que se recomienda a futuros investigadores el uso de estas ventajas virtuales donde se puede desarrollar grandes sistemas de computación sin necesidad de contar con una computadora física de características altas.

Mientras se consiga mayor cantidad de información para entrenar la red es mejor, es recomendable buscar set de datos grandes y de buena calidad para obtener mejores resultados.

El prototipo no es capaz de tomar la radiografía, por lo que se invita a futuros investigadores diseñar el sistema completo que permita la captura de imágenes de rayos X, el interfaz usuario máquina y un despliegue gráfico de la zona dañada de los pulmones.

La arquitectura de la red neuronal convolucional diseñada puede ser reentrenada y utilizada para diferentes tipos de aplicaciones que tengan una similitud con el objetivo del presente proyecto, es decir diagnóstico de otro tipo de enfermedades a partir de imágenes o reconocimiento de objetos en imágenes.

MATERIAL ANEXO Y COMPLEMENTARIO.

Código desarrollado en lenguaje Python:

```python
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

#vectores para guardar datos
categorias =[]
etiquetas = []
imagenes = []
categorias = os.listdir('/content/drive/My Drive/1040/train')
print(categorias)
```

```python
#cargar imagenes del drive
x=0
for directorio in categorias:
    for imagen in os.listdir('/content/drive/My Drive/1040/train/'+directorio):
        img = Image.open('/content/drive/My Drive/1040/train/'+directorio+'/'+imagen).convert('RGB').resize((224,224))
        img = np.asarray(img)
        imagenes.append(img) #agregamos imagen tamaño 224x224x3
        etiquetas.append(x)
        #cada etiqueta se va cargando de forma ordenada depende dl vector categorias normal=2 bacteria=1 virus=0
    x+=1
```

```python
#convertir a arreglos
etiquetas=np.asarray(etiquetas)
imagenes = np.asarray(imagenes)
print(imagenes.shape)
print(etiquetas)

#desorden de las imagenes y sus etiquetas con la misma seed
np.random.seed(23)
np.random.shuffle(imagenes)
np.random.seed(23)
np.random.shuffle(etiquetas)
print(etiquetas)
```

```python
#visualizacion de imagenes con su etiqueta
import matplotlib.pyplot as plt
import cv2
i=1
plt.figure(figsize=(20,20))
for imagen, eti in zip(imagenes[10:22], etiquetas[10:22]):
  imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
  plt.title(eti)
  plt.subplot(4,3,i)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(imagen, cmap='gray')
  i=i+1
```

```
#PREPROCESAMIENTO
#convertir las imagenes a escala de grises
ima=[]
for imagen in imagenes:
  imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
  imagen = imagen.reshape(224,224,1)
  ima.append(imagen)

#Normalizacion de los pixeles
ima= np.asarray(ima)
print(ima.shape)
X = ima
y = etiquetas
X = np.asarray(X).astype(float) / 255
```

```
#AUMENTO DE DATOS
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=23, # rotacion aleatoria imagen
    width_shift_range=0.1, # mover a los lados de 0 a 1
    height_shift_range=0.1, # arriba y abajo
    #shear_range = 2, #la imagen se incline
    zoom_range= [0.8, 1.1], #se achica 80% o se agranda 110%
    vertical_flip = False, #rotacion vertical u horizontal_flip
    horizontal_flip=False #True
)

datagen.fit(X)
plt.figure(figsize=(20,8))

for imagen, etiqueta in datagen.flow(X, y, batch_size=10, shuffle=False):
  for i in range(10):
    plt.subplot(2,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(imagen[i].reshape(224,224), cmap='gray')
  break
```

```
#AUMENTO DE DATOS EN LOTES DE 32
data_gen_entrenamiento = datagen.flow(x_entrenamiento, y_entrenamiento, batch_size=32)

from tensorflow.keras.callbacks import TensorBoard
```

```
#ARQUITECTURA DE LA RNC
modeloRNC = tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(64, (4,4), activation='relu',input_shape=(224, 224, 1), use_bias=True, padding="same", strides=2),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (4,4), activation='relu',use_bias=True, padding="same", strides=2),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(256, (4,4), activation='relu',use_bias=True, padding="same", strides=2),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(512, (4,4), activation='relu',use_bias=True, padding="same", strides=2),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.8),

    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.3),

    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
#OPTIMIZADOR Y FUNCION DE COSTE
modeloRNC.compile(optimizer='adam',
                  loss= 'sparse_categorical_crossentropy',
                  metrics = ['accuracy']
)
```

```
#Separació manual de datos de entrenamiento y validación
len(X)*.85 #3060
len(X)-3060 #630

x_entrenamiento = X[:3100]
x_validacion = X[3100:]

y_entrenamiento = y[:3100]
y_validacion = y[3100:]
```

```
#ENTRENAMIENTO DE LA RNC
tensorboardRNC = TensorBoard(log_dir='/content/graficos/RNC')

modeloRNC.fit(data_gen_entrenamiento,
                batch_size=32,
                epochs=160,
                validation_data=(x_validacion, y_validacion),
                steps_per_epoch = int(np.ceil(len(x_entrenamiento)/float(32))),
                validation_steps = int(np.ceil(len(x_validacion)/float(32))),
                callbacks= [tensorboardRNC])
```

```
#VISUALIZACION DE RESULTADOS
%load_ext tensorboard

%tensorboard --logdir graficos
```

BIBLIOGRAFIA Y REFERENCIAS.

Cañadas, R. (2021). *Entropía de Shannon.* Obtenido de https://abdatum.com/ciencia/entropia-shannon

Diederik P. Kingma y Jimmy Lei Ba. (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. *conference paper at ICLR 2015.*

Dot CSV. (2019). *¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial.* Obtenido de https://www.youtube.com/watch?v =A6FiCDoz8_4

Dot CSV. (2019). *¿Qué es una Red Neuronal? Parte 1: La Neurona*. Obtenido de https://www.youtube.com/watch?v=uwbHOpp9xkc&list=PLOgd76BhmcC_E2RjgIIJZd1DQdYHcVf0&index=7.

García Serrano, A. (2019). *Visión artificial con redes convolucionales (CNN)*. Obtenido de https://www.ellaberintodefalken.com/2019/10/vision-artificial-redes-convolucionales-CNN.html

Hidalgo Barrientos, M., Hayes Ortiz, B., Delgadillo Vera, I., & Goyo Escalona, M. (27 de octubre de 2021). *Deep learning aplicado para la detección de hemorragias y tumores cerebrales.* Obtenido de https://brapci.inf.br/index.php/res/download/169547.

Jimenez, C. (18 de enero de 2023). *La IA no te va a quitar tu trabajo, te lo quitará la persona que sepa usar la IA para desarrollar mejor tu trabajo.* Obtenido de LinkedIn: https://es.linkedin.com/pulse/la-ia-te-va-quitar-tu-trabajo-lo-quitará-persona-que-jimenez-cabrera

Kermany D., G. M. (22 de febrero de 2018). *Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning.* Obtenido de https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5#relatedArticles

Kumar, S. (2020). *Redes neuronales de convolución en pocas palabras Guía de la A a la Z.* Obtenido de https://www.herevego.com/redes-neuronales-de-convolucion/

Mayo Foundation for Medical Education and Research . (s.f.). *Chest X-ray showing pneumonia*. Obtenido de https://www.mayoclinic.org/diseases-conditions/pneumonia/multimedia/chest-x-ray-showing-pneumonia/img-20005827

Morandín-Ahuerma. (enero de 2023). *What is artificial intelligence?* Obtenido de https://philpapers.org/archive/MORQEI-2.pdf

OpenAI. (2022). *DALL-E2.* Obtenido de https://openai.com/dall-e-2

OpenAI. (14 de Marzo de 2023). *GPT-4 Technical Report.* Obtenido de https://arxiv.org/pdf/2303.08774.pdf

Organización Mundial de La Salud. (2022). *Neumonía infantil.* Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/pneumonia

P. Kingma, D., & Lei Ba, J. (2015). *ADAM: A Method for Stochastic Optimization.* Obtenido de https://arxiv.org/pdf/1412.6980.pdf

Rajpurkar, P., Irvin , J., Zhu, K., Yang, B., Mehta, H., Duan, T., . . . Y. Ng, A. (25 de Diciembre de 2017). *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays.* Obtenido de https://arxiv.org/pdf/1711.05225.pdf

Ringa Tech. (2021). *Reconocimiento de imágenes con IA Convoluciones y filtros*. Obtenido de https://www.youtube.com/watch?v=AwTH_0yW9_I

Rumelhart, D., E. Hinton, G., & J. Williams, R. (1 de octubre de 1986). *Learning representations by back-propagating errors.* Obtenido de http://www.cs.utoronto.ca/~hinton/absps/naturebp.pdf

SEDES. (2023). *SEDES REPORTA INCREMENTO DE CASOS DE NEUMONÍA.* SEDES. La Paz: SEDES LA PAZ. Obtenido de https://www.sedeslapaz.gob.bo/sedes-reporta-incremento-de-casos-de-neumonia/

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (junio de 2014). *Dropout: A Simple Way to Prevent Neural Networks from.* Obtenido de https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Viceministerio de Comunicación de Bolivia. (12 de junio de 2020). *Viceministerio de Comunicación de Bolivia*. Obtenido de https://comunicacion.gob.bo/?q=20200612/29720

Zhang, K. (30 de noviembre de 2021). *Elespanol.com*. Obtenido de https://www.elespanol.com/omicrono/tecnologia/20190213 /medicina-inteligencia-artificial-capaz-diagnosticar- mejor- medicos/ 375963771_0.html

GLOSARIO DE TÉRMINOS.

- **Clúster**: aglomerado, grupo o conjunto.
- **Consolidación lobar**: es el patrón radiográfico más común en la radiografía de tórax que demuestra presencia de neumonía.
- **Epidemiología**: es el estudio de la frecuencia y distribución de los eventos de salud y de sus determinantes en las poblaciones humanas.
- **Esputo**: también conocido como flema, es un tipo de mucosidad espesa que se produce en los pulmones.
- **Estetoscopio**: Instrumento médico que sirve para explorar los sonidos producidos por los órganos de las cavidades del pecho y del abdomen.
- **Fúngica**: Perteneciente o relativo a los hongos.
- **Kaggle**: Es una comunidad en línea de científicos de datos y profesionales del aprendizaje automático, permite a los usuarios encontrar y publicar conjuntos de datos, explorar y crear modelos en un entorno de ciencia de datos basado en la web.
- **Oftalmología**: Parte de la medicina que estudia el ojo y se ocupa de sus enfermedades.
- **Opacificación:** Proceso que genera un aumento de la densidad o capacidad de atenuación de los rayos X.
- **OpenAI**: Es una empresa de investigación e implementación de IA. Cuya misión es garantizar que la inteligencia artificial general beneficie a toda la humanidad.
- **Pediátrico**: Se considera un paciente pediátrico desde que nace hasta su adolescencia.
- **Pleurítico**: manifestación de dolor en zonas pulmonares.
- **Rural**: asentamientos o pueblos pequeños los cuales tienen una baja densidad de población.
- **Tórax:** Área del cuerpo entre el cuello y el abdomen.

ACRÓNIMOS.

GAN: Generative Adversarial Network.

GPT: Generative Pre-trained Transformer.

IRA: Infección Respiratoria Aguda.

IA: Inteligencia Artificial.

ML: Machine Learning.

NLP: Natural Language Processing.

RN: Red Neuronal.

RNC: Red Neuronal Convolucional.

SEDES: Servicio Departamental de Salud.

2024-TTES-334-D-1

DIRECCIÓN DE DERECHO DE AUTOR
Y DERECHOS CONEXOS
RESOLUCIÓN ADMINISTRATIVA NRO. 1-1095/2024
La Paz, 10 de abril de 2024

VISTOS:

La solicitud de Inscripción de Derecho de Autor presentada en fecha **03 de abril de 2024**, por **MARTÍN MIGUEL JURADO CAMACHO** con **C.I. Nº 7137581 TJ**, con número de trámite **DA 578/2024**, señala la pretensión de inscripción del Proyecto de Grado titulado: **"Diseño de un Prototipo de Sistema Neuronal de Diagnóstico y Detección de Neumonía Pulmonar."**, cuyos datos y antecedentes se encuentran adjuntos y expresados en el Formulario de Declaración Jurada.

CONSIDERANDO:

Que, en observación al Artículo 4º del Decreto Supremo Nº 27938 modificado parcialmente por el Decreto Supremo Nº 28152 el "*Servicio Nacional de Propiedad Intelectual SENAPI, administra en forma desconcentrada e integral el régimen de la Propiedad Intelectual en todos sus componentes, mediante una estricta observancia de los regímenes legales de la Propiedad Intelectual, de la vigilancia de su cumplimiento y de una efectiva protección de los derechos de exclusiva referidos a la propiedad industrial, al derecho de autor y derechos conexos; constituyéndose en la oficina nacional competente respecto de los tratados internacionales y acuerdos regionales suscritos y adheridos por el país, así como de las normas y regímenes comunes que en materia de Propiedad Intelectual se han adoptado en el marco del proceso andino de integración*".

Que, el Artículo 16º del Decreto Supremo Nº 27938 establece "*Como núcleo técnico y operativo del SENAPI funcionan las Direcciones Técnicas que son las encargadas de la evaluación y procesamiento de las solicitudes de derechos de propiedad intelectual, de conformidad a los distintos regímenes legales aplicables a cada área de gestión*". En ese marco, la Dirección de Derecho de Autor y Derechos Conexos otorga registros con carácter declarativo sobre las obras del ingenio cualquiera que sea el género o forma de expresión, sin importar el mérito literario o artístico a través de la inscripción y la difusión, en cumplimiento a la Decisión 351 Régimen Común sobre Derecho de Autor y Derechos Conexos de la Comunidad Andina, Ley de Derecho de Autor Nº 1322, Decreto Reglamentario Nº 23907 y demás normativa vigente sobre la materia.

Que, la solicitud presentada cumple con: el Artículo 6º de la Ley Nº 1322 de Derecho de Autor, el Artículo 26º inciso a) del Decreto Supremo Nº 23907 Reglamento de la Ley de Derecho de Autor, y con el Artículo 4º de la Decisión 351 Régimen Común sobre Derecho de Autor y Derechos Conexos de la Comunidad Andina.

Que, de conformidad al Artículo 18º de la Ley Nº 1322 de Derecho de Autor en concordancia con el Artículo 18º de la Decisión 351 Régimen Común sobre Derecho de Autor y Derechos Conexos de la Comunidad Andina, referentes a la duración de los Derechos Patrimoniales, los mismos establecen que: "*la duración de la protección concedida por la presente ley será para toda la vida del autor y por 50 años después de su muerte, a favor de sus herederos, legatarios y cesionarios*"

Que, se deja establecido en conformidad al Artículo 4º de la Ley Nº 1322 de Derecho de Autor, y Artículo 7º de la Decisión 351 Régimen Común sobre Derecho de Autor y Derechos Conexos de la Comunidad Andina que: "*...No son objeto de protección las ideas contenidas en las obras literarias, artísticas, o el contenido ideológico o técnico de las obras científicas ni su aprovechamiento industrial o comercial*"

Que, el artículo 4, inciso e) de la ley N° 2341 de Procedimiento Administrativo, instituye que: "*... en la relación de los particulares con la Administración Pública, se presume el principio de buena fe. La confianza, la cooperación y la lealtad en la actuación de los servidores públicos y de los*

ciudadanos ...", por lo que se presume la buena fe de los administrados respecto a las solicitudes de registro y la declaración jurada respecto a la originalidad de la obra.

POR TANTO:

El Director de Derecho de Autor y Derechos Conexos sin ingresar en mayores consideraciones de orden legal, en ejercicio de las atribuciones conferidas.

RESUELVE:

INSCRIBIR en el Registro de Tesis, Proyectos de Grado, Monografías y Otras Similares de la Dirección de Derecho de Autor y Derechos Conexos, el Proyecto de Grado titulado: **"Diseño de un Prototipo de Sistema Neuronal de Diagnóstico y Detección de Neumonía Pulmonar."** a favor del autor y titular: **MARTÍN MIGUEL JURADO CAMACHO** con **C.I. Nº 7137581 TJ**, quedando amparado su derecho conforme a Ley, salvando el mejor derecho que terceras personas pudieren demostrar.

Regístrese, Comuníquese y Archívese.

CASA/lm

Firmado Digitalmente por:
Servicio Nacional de Propiedad Intelectual - SENAPI
CARLOS ALBERTO SORUCO ARROYO
DIRECTOR DE DERECHO DE AUTOR Y DERECHOS CONEXOS
LA PAZ - BOLIVIA

Firma:

GBrAh6Pl8Gi17K

PARA LA VALIDACIÓN DEL PRESENTE DOCUMENTO INGRESAR A LA PÁGINA WEB **www.senapi.gob.bo/verificacion** Y COLOCAR CÓDIGO DE VERIFICACIÓN O ESCANEAR CÓDIGO QR.

Autor: Martín Miguel Jurado Camacho.

Correo Electrónico: martijurado7@gmail.com

Número de celular: +591 78702018